Univerza v Ljubljani

Fakulteta za računalništvo in informatiko

University of Ljubljana

Faculty of Computer and Information Science

Zbornik

Digitalna forenzika

Seminarske naloge, 2017/2018

Ljubljana, 2018

Zbornik

Digitalna forenzika, Seminarske naloge 2017/2018

Editors: Andrej Brodnik, David Klemenc, študenti

Ljubljana : Univerza v Ljubljani, Fakulteta za računalništvo in informatiko 2018.

Kazalo / Contents

1 Uvod / Introduction

Digitalna forenzika je veja forenzične znanosti, ki zajema obnovo in preiskavo gradiva najdenega v digitalnih napravah in je pogosto v povezavi z računalniškim kriminalom. Izraz digitalne forenzike je bil prvotno uporabljen kot sinonim za računalniško forenziko, vendar se je razširil, da bi zajel preiskavo vseh naprav, ki lahko shranjujejo digitalne podatke. Korenine lahko zasledimo v osebni računalniški revoluciji v poznih sedemdesetih in zgodnjh osemdesetih letih. V devetdesetih je razvoj discipline potekal brez prave organizacije, dokler se niso v 21. stoletju pojavile nacionalne smernice.

Digitalne forenzične preiskave imajo različne naloge. Najpogostejše so podpirati ali ovreči hipotezo pred kazenskimi ali civilnimi sodišči. Kriminalni primeri vključujejo domnevno kršitev zakonov, ki jih določa zakonodaja, kot so naprimer umori, kraje in napadi na osebo. Civilni primeri pa se ukvarjajo z zaščito pravic in lastnine posameznikov (pogosto povezani z družinskimi spori), lahko pa se tudi ukvarjajo s pogodbenimi spori med gospodarskimi subjekti, pri katerih se vključi oblika digitalne forenzike, ki se imenuje elektronsko odkritje.

V zborniku so zbrane seminarske naloge študentov magistrskega študija na Fakulteti za računalništvo in informatiko Univerze v Ljubljani 2017/2018. V okviru predmeta Digitalna forenzika je vsaka skupina študentov izbrala en članek, ki je služil kot izhodišče za seminarsko delo.

Članki so bili izbrani iz šestih raziskovalnih področij: metodologija, forenzika mobilnih naprav, diskovna forenzika, forenzika pomnilnika, analiza signalov in forenziko interneta stvari (IoT ang. *Internet of Things*) ter članek, ki ni spadal v nobeno od prej omenjenih kategorij.

Pri področju metodologije se članki dotikajo tematik o časovni zahtevnosti digitalnih preiskav, horizontalnega skaliranja presikave s pomočjo paralalnega procesiranja ter problematik pri varstvu osebnih podatkov.

Forenzika mobilnih naprav je najpodrobneje predstavljeno področje, saj se ravno na tem področju v zadnjem času intenzivno razvija forenzično opremo in prepreke zanjo. Seminarske naloge predstavijo orodja in metode za ponarejanje SMS (ang. *Short Message Service*) sporočil, problematiko dokazovanja integritete podatkov s zgoščevalnimi funkcijami (ang. *hash functions*) in forenzične preiskave CDR (ang. *call detail record*) podatkov mobilnih operaterjev. Pregledajo tudi preiskave dveh najpopularnejših platform pametnih telefonov - Android in iOS ter analizirajo programsko opremo za pridobivanje in manipulacijo GPS podatkov.

Na področju diskovne forenzike pregledamo nova orodja za avtomatično popravljanje pokvarjenih datotek, metode analize dedupliciranih datotečnih sistemov ter forenzično analizo ext4 datotečnega sistema.

V sklopu forenzike pomnilnika seminarske naloge zajamejo pregled novih metod pregledovanja spomina uporabnikovih programov na sistemih Linux, analizo pomnilnika javanskega navideznega stroja (JVM ang. *java virtual machine*) ter pregled orodja *Gaslight*, ki služi za testiranje programske opreme namenjene preiskovanju pomnilnika.

Pri forenzični analizi signalov so obravnavani tako zvočni signali in sicer problem ujemanja frekvence električnega omrežja (ENF ang. *electrical network frequency*) vzorcev v sklopu ocenjevanja časa snemanja zvočnih posnetkov kot signali, ki jih pridobimo z zajemanjem vzorca tipkanja (ang. *keystroke dynamics*), s pomočjo katerega lahko z veliko verjetnostjo določimo spol uporabnika in tudi druge lastnosti.

V sklopu internet stvari (ang. *Internet of Things*) pregledamo arhitekturo PCL-jev (ang. *Programmable Logic Controllers*) in njihove ranljivosti ter metode udiranja. Pregledamo tudi metode digitalne forenzike, ki se nanašajo na ekosistem Amazon Alexa in pri tem spoznamo nov pristop, ki združuje cloud-native forenziko in forenziko na strani odjemalca.

V sklopu razno pa seminarska naloga predstavi odprtokodno orodje *DRone Open source Parser* (DROP), ki je namenjeno lastniškim datotekam DAT pridobljenih iz notranjega pomnilnika brezpilotnega letala (DJI Phantom III).

Ta zbornik združuje vse končne seminarske naloge, ki so bile izdelane v študijskem letu 2017/2018. Namenjen je vsem, ki jih zanima področje digitalne forenzika ali pa zgolj eno ali več predstavljenih področij.

Ljubljana, 2018 David Klemenc

2 Povzetki / Summaries

2.1 Programming, investigation and documentation in digital investigation

This paper discuss the amount of effort and time required to hold a Digital investigation in terms of programming, investigation also the documentation and logging of what is done. The paper follow the experiment discussed in the paper "Do digital investigators have to program? A controlled experiment in digital investigation". The experiment research a lot of aspects regarding the process of a investigating a dig-ital involved crime, those aspects differ from the relation between amount of effort in terms of time invested in the investigation with the quality of the results, to the need of programming skills or the implementation of any programs through out the carrying on of a digital investigation. The experiment was held on 39 students who were split in 10 groups all of are computer science graduate students.

2.2 SCARF: Skaliranje digitalnega forenzičnega procesiranja z uporabo vsebniških tehnologij v oblaku

V zadnjih letih je zmogljivost računalnikov močno zrasla. Posledično so se povečale tudi kapacitete podatkovnih nsilcev oz. diskov. Temu razvoju bodo morali slediti tudi forenzični postopki, da se bodo lahko prilagodili na zmogljivejše računalniške vire.
S tem namenom so bile razvite nove programske rešitve, ki omogočajo odprto in fleksibilno integracijo že obstoječih orodij, poleg tega pa podpirajo skalabilnost in prilagodljivost na zahtevnejše procesiranje podatkov. Eno od tovrstnih ogrodij je ogrodje SCARF. Uporablja princip vsebnikov (ang. containerization) ter s tem omogoča vzporedno procesiranje podatkov, ki se skalira glede na zahtevnost dane naloge. Poleg tega omogoča tudi enostavno dodajanje razširitev.

2.3 Review of the paper: Availability of datasets for digital forensics – And what is missing

This paper is a review of the paper Availability of datasets for digital forensics and what is missing, with an overview of the wider area, the referenced papers and concluded with our own opinion of the matter. There are many challenges for researchers that require datasets in the field of Digital Forensics and we try to identify them and offer an overview of the whole situation based on the original paper. We discuss the availability of repositories, the problem with sharing data and we present the findings and results.

2.4 Selektivno brisanje podatkov

Vse več forenzičnih preiskav dandanes vključuje tudi zaseg digitalnih naprav. Večje ali manjše količine podatkov na zaseženih napravah so zasebne narave. Z vse strožjimi zakoni o varstvu osebnih podatkov in zasebnega življenja se digitalni forenziki v Evropi in po svetu srečujejo s problemom kako takšne podatke zares uničiti, obenem pa s tem ne kvariti verodostojnosti dokazov. Postavimo se tudi na drugi breg, kako bi lahko orodja za selektivno brisanje podatkov uporabili za anti-forenzično delovanje. Recimo da je bil vohun razkrinkan in se mora hitro znebiti občutljivih podatkov. Pojavi se problem kako se podatkov znebiti na nesumljiv način. Nazadnje še predstavimo primer implementacije sistema za selektivno brisanje podatkov na NTFS datotečnih sistemih. Ta je bil implementiran za skupek orodij Digital Forensic Framework.

2.5 Tools and methods for falsification of SMS messages

In this paper we present problem of presenting SMS messages in court as a digital evidence. SMS messages are still one of the most used forms of communication, therefore exists possibility to be present in court as evidence. This article consists of several parts. First, we describe how SMS messages work regardless of platform and the usage of public SMS gateways. Then we describe how SMS messages were falsified in the past (around 2010) on Nokia phone. Lastly,we present contemporary processing on modern devices like Android and iOS with an experiment of falsification of SMS on Android device.

2.6 Hash tehnike za forenzične preiskave mobilnih naprav

Raziskave, opravljene na National Institute of Standards and Technology, so pokazale, da so hash vrednosti notranjih pomnilnikov mobilnih naprav spremenljive pri opravljanju back-to-back pregleda. Hash vrednosti so koristne pri zagotavljanju izvedencem možnosti filtriranja znanih podatkovnih datotek, ujemanja podatkovnih objektov na različnih platformah in

dokazati, da integriteta podatkov ostaja nedotaknjena. Raziskava, izvedena na Univerzi Purdue, je primerjala znane hash vrednosti z izračunanimi vrednostmi za podatkovne predmete naložene na mobilne naprave z različnimi metodami prenosa podatkov. Medtem ko so bili rezultati za večino preizkusov enotni, so bile hash vrednosti, ki so bile izračunane za podatkovne predmete, prenesene prek storitve za večpredstavnostna sporočila (MMS), spremenljive.

2.7 Forenzična preiskava s pomočjo podatkov mobilnih operaterjev

V tem poročilu bo govora o CDR (call detail record) oziroma o podrobnostih klica, ki se beležijo ob uporabi mobilnih telefonov v današnjem času. Pogledali si bomo, kaj ti podatki vsebujejo, kaj je njihov namen in tudi kako se pridobivajo in hranijo. Poročilo je nastalo na podlagi kazenskega primera, kjer so bili prav ti podatki uporabljeni na sodišču in so dokazali, da je oseba obtožena kaznivega dejanja krivo pričala glede svoje lokacije.

2.8 Forenzika Androida

Pametne mobilne naprave so postale pravi superračunalniki. Te nam pomagajo pri vsakodnevnih opravilih ter pri tem beležijo ogromne količine podatkov, ki lahko postanejo velik vzvod v primeru, da pristanejo v napačnih rokah. Cilj članka je bralcu pokazati kako enostavno najdemo nekatere pomembne podatke in ga s tem prepričati kako pomembno je imeti ustrezno zavarovano mobilno napravo. Osredotočimo se na naprave, ki poganjajo operacijski sistem Android. Slednjega na kratko tudi opišemo. Srednji del članka je posvečen povzetku Android Forensics: Simplifying Cell Phone Examinations ter predstavitvi njihovih metod preiskovanja in analize. Na koncu opišemo tudi metode in rezultate našega poskusa preiskovanja in analize naprave Nexus 4. Najdeni podatki so nas nedvomno presenetili.

2.9 Ocena integritete podatkov pri mobilni forenziki s spremljanjem dogodkov

Zaradi širokega spektra storitev, ki jih mobilni telefoni ponujajo, postajajo vedno bolj pomembno orodje vsakdanjega življenja ljudi. Zato bi lahko delovali kot temeljne priče ali preprosto kot vir informacij pri podpiranju preiskav številnih kaznivih dejanj, ki niso omejena le na digitalni kriminal. Trenutna forenzična pridobitev in analiza naprav temelji na orodjih za izvajanje daljinskega upravljanja, pri katerih se uporablja forenzično delovno postajo za zaseg z vstavljanjem kode v mobilno napravo. Iz tega sledi, da je karakterizacija spoštovanja integritete še vedno težavna in potrebuje poglobljeno raziskavo. Avtorji članka so predstavili nov pristop za oceno spoštovanja integritete v zvezi z orodji za pridobivanje informacij. Rezultati poskusov kažejo primernost predlagane strategije.

2.10 iPhone Forensics

iPhone is one of the most popular mobile devices today and therefore it is logical that it can represent the essential part in an investigation, since vital information from these devices can make critical part of investigative evidences. The challenge is the extraction of data of forensic value such as e-mail messages, text and multimedia messages, calendar events, browsing history, GPRS locations, contacts, call history, voicemail recording, etc.

2.11 Expanding the Potential for GPS Evidence Acquisition

This paper was written based on the paper with the same title which looks into GPS data for investigating purposes. The number of GPS devices and their capabilities have increased immensely during last years which gives the investigators more tools in the forensic procedure and gives criminals more ways to manipulate data in order to mislead the investigators. We will write about GPS network, devices related to GPS, what kind of software is used to gain quality information and possible information collected during an forensic investigation involving GPS receivers.

2.12 Force Open: Lightweight black box file repair

This paper is a summary of a novel approach for automatic repair of corrupted files in study under review. Study presents a lightweight approach that modifies execution of a file viewer, forcing it to open corrupted files. File viewer is referred as a black box that makes this technique file format independent and only requires access to a program binary. According to original authors' results, rate of successfully opened corrupted files in combination with other existing file repair tools was increased. Approach was implemented and evaluated for PNG, JPG and PDF files.

2.13 Forenzična analiza dedupliciranih datotečnih sistemov

Deduplikacija razdeli datoteke na fragmente, ki so shranjeni v skladišču za drobce (ang. chunck repository.) Drobci, ki so sorodni za večje število datotek, so shranjeni le enkrat. S perspektive računalniške forenzike so podatki z naprav, ki uporabljajo deduplikacijo težko obnovljeni, potrebno je po- sebno znanje, kako tehnologija deluje. Proces deduplikacije spremeni celotno datoteko na organiziran niz fragmentov. Do nedavnega je bila ta tehnologija uporabljena le v podatkovnih središčih, kjer je bila uporabljena za zmanjševanje porabe prostora rezervnih kopij. Zdaj je ta dostopna v odprtokodnih paketih kot je OpenDedup, ali pa kot sistemski dodatek operacijskega sistema. Tak primer je Microsoft z dodatkom v Windows 10 Technical Preview. Orodja, s katerimi se izvajajo preiskave, morajo biti izpopolnjena, da zaznajo, analizirajo in obnovijo vsebino dedupliciranih datotečnih sistemov. Deduplikacija namreč doda dodaten sloj k dostopu do podatkov. Ta sloj mora biti raziskan, da je zaseg kot nadaljnja analiza izvedena pravilno. V tem članku je predstavljena deduplikacija, ter uporaba v OpenDedup ter na operacijskem sistemu Windows 2012.

2.14 Advanced forensic Ext4 inode carving: summar

Many widely-used filesystems (NTFS, FAT, Ext3) are well-researched in terms of digital forensics and data recovery, but this does not apply to Ext4, a successor in the family of Ext filesystems. Due to some new functionalities of Ext4 and compatibility breaking, a novel approach for file carving had to be developed. The advantages of this approach include its ability to restore files even in the case of a corrupted superblock. This article gives a summary of the original paper and its outcomes, also with an introduction to Ext4 filesystem and Ext family included.

2.15 Linux memory forensics: Dissecting the user space process heap

This work is an overview of a paper on a new method of performing memory forensics on the heap of Linux user space programs. We present common structures found in the glibc implementation of the process heap and their general organization. We summarize how the knowledge of these structures can help locate program data in memory, confirm none of it was overlooked and separate it from heap's meta- data and data belonging to other programs. We also give a general overview of Rekall, the framework used in exploring the memory, test out the new method and verify the reproducibility of the source paper's results on both old and new versions of analysed software.

2.16 Analiza pomnilnika z uporabo generacijskega čistilca pomnilnika

Analiza glavnih pomnilnikov lahko predstavlja pomemben del forenzične analize. Zaradi razmaha programskih jezikov, ki tečejo v navideznih strojih, pa je potrebno popraviti, oziroma na novo razviti orodja, ki poizkušajo razbrati informacije, vsebovane v njih.
Ta članek se osredotoča na analizo pomnilnika navideznih strojev - natančneje Javanskega navideznega stroja (JVM) HotSpot in pripadajočega čistilca pomnilnika (ang. garbage collector - GC). Kljub temu, da je bil nek podatek v programu označen za izbris, pa lahko v pomnilniku ostane še precej časa, saj HotSpot ne briše podatkov za seboj, vendar jih le kopira. S podrobnejšo analizo se je izkazalo, da lahko zaradi tega še vedno pridemo do podatkov, za katere najprej sklepamo, da niso več dostopni.
Ta članek se posveti analizi pomnilnika navideznega stroja HotSpot, vendar bi ga lahko posplošili tudi za druge podobne stroje, kot sta Microsoftov .Net in Googlov V8 JavaScript Engine.

2.17 Crashing programs for fun and profit

We review an efficient architecture for testing memory forensics applications with fuzzing, called Gaslight. We present different ways of fuzzing, compare them to Gaslight, present its authors' results, then repeat the experiment. We also present some problems and propose several possible enhancements to the architecture.

2.18 Keystroke dynamics features for gender recognition

In digital forensics, one often finds himself in a position where the ability to profile the computer user may simplify the task of finding a suspect. Most of the research in the field focuses on recognizing the gender or age which are two of the most informative characteristics and usually the first ones a digital forensic wants to know. The ways to do so vary from using complex ones such as voice recordings, images, signatures to fairly simple ones like the way a person types. This field of study is called keystroke dynamics. Authors of the reference paper chose to predict a person's gender based on keystroke dynamics, since this is, as opposed to the pictures, a non-intrusive method. They assembled a dataset, recording users during daily computer usage, calculated features and lowered the dimensionality of data and finally trained a few of

the most popular classifiers for this binary classification task. Using a radial-basis function network (RBFN), they achieve the highest accuracy reported in the field to date.

2.19 Ocenjevanje časa snemanja zvočnih posnetkov

To delo obravnava problem ujemanja ENF (frekvenca električnega omrežja, ang. electrical network frequency) vzorcev v sklopu ocenjevanja časa snemanja zvočnih posnetkov. V delu je po navdihu vizualne primerljivosti predlagan in opisan nov kriterij podobnosti (bitna podobnost) za merjenje podobnosti med dvema ENF signaloma. Predstavitvi kriterija sledi opis iskalnega sistema, ki najde najboljše ujemanje za določen testni signal ENF v velikem obsegu iskanja na referenčnih ENF podatkih. Z empirično primerjavo z drugimi priljubljenimi kriteriji podobnosti je v delu dokazano, da je predlagana metoda bolj učinkovita od najsodobnejših tehnik. Na primer, v primerjavi z nedavno razvitim algoritmom DMA omenjena metoda doseže 86.86% nižjo relativno napako in je za približno 45-krat hitrejša od DMA. Na koncu je predstavljena strategija preizkusa edinstvenosti za pomoč človeškim ocenjevalcem pri zagotavljanju natančnih odločitev, posledično je metoda praktično uporabna v forenziki.

2.20 Forensics of Programmable Logic Controllers

A programmable logic controller in many regards bears resemblance to a general-purpose computer or a microcontroller, but possesses important characteristics that make its significance in industrial automation much more prominent. This shows in the fact that most critical infrastructure today heavily relies on PLCs and other industrial control systems. Regardless of their value, little concern has been given to the security of said systems in the past. This is due to the fact that initially many devices used in an industrial automation, along with PLCs, were meant to be used in isolation – disconnected from other devices in the industrial environment. As industrial control systems evolved, they have started to rely heavily on the network and use of internet-based standards to share valuable data within large corporate networks. Hence, they have become vulnerable to a completely new set of exploits, that were traditionally used to target computers in a network. This has changed for the better over the years in which the industrial automation has become widespread. In this work we give a primer on PLCs and their architecture, an overview of possible vulnerabilities and ways of intrusion and forensic challenges associated with them. Furthermore, we characterize a particular PLC and give insights into its intricacies and inner workings. Additionally, the proprietary GE-SRTP protocol is presented and evaluated as a means to obtain data from the device in forensic investigation.

2.21 Pristopi digitalne forenzike za ekosisteme Amazon Alexa

Pametni zvočnik Amazon Echo je zelo pomemben za inteligentno virtualno pomočnico Alexo, ki je bila razvita s strani Amazon Lab126. Amazon Echo posreduje glasovne ukaze do Alexe, katera se sporazumeva z obilico združljivih naprav interneta stvari (ang. Internet-of-Things - IoT) in aplikacij drugih razvijalcev. IoT naprave, kot je Amazon Echo, ki so stalno vklopljene in povsod navzoče, so lahko zelo dober vir potencialnih digitalnih dokazov. Za podporo digitalnim preiskavam je pomembno razumevanje kompleksnega oblačnega ekosistema, ki omogoča uporabo Alexe. V članku so obravnavane metode digitalne forenzike, ki se nanašajo na ekosistem Amazon Alexa. Glavni del članka je namenjen novemu pristopu digitalne preiskave, ki združuje forenzično analizo artefaktov v oblagu (ang. *forensic analysis of cloud-native artifacts* in forenziko na strani odjemalca. Predstavljeno je tudi orodje CIFT (Cloud-based IoT Forensic Toolkit), ki omogoča identifikacijo, pridobitev in analizo tako cloud-native artefaktov v oblaku, kot odjemalsko-centričnih podatkov na lokalnih napravah.

2.22 DROP (DRone Open source Parser): Forenzična analiza modela DJI Phantom III

V sklopu tega dela smo analizirali članek DROP(DRone Open source Parser) your drone: Forensic analysis of the DJI Phantom III avtorjev Clark et al.
Brezpilotno letalo DJI Phantom III je bilo v letih 2016 in 2017 že uporabljeno pri zlonamernih dejavnostih. V času pisanja analiziranega članka je bilo podjetje DJI proizvajalec z največjim tržnim deležem na področju brezpilotnih letal. Avtorji Clark et al. so predstavili forenzično analizo modela DJI Phantom III ter implementirali razčlenjevalnik za strukture datotek, ki jih hrani preiskovano brezpilotno letalo.
V obravnavanem članku so avtorji predstavili odprtokodno orodje DRone Open source Parser (DROP), ki je namenjeno lastniškim datotekam DAT pridobljenih iz notranjega pomnilnika brezpilotnega letala. Analizirani članek vsebuje predhodne ugotovitve o datotekah TXT, ki se nahajajo na mobilni napravi, ki upravlja in nadzoruje brezpilotno letalo. Z izločanjem podatkov iz nadzorovalne mobilne naprave in brezpilotnega letala so avtorji Clark et al. korelirali podatke ter na podlagi pridobljenih podatkov povezali uporabnika z določeno napravo. Poleg tega so rezultati analiziranega članka

pokazali, da je najboljši mehanizem za forenzično pridobivanje podatkov iz brezpilotnega letala, ročna odstranitev kartice SD.
Ugotovitve avtorjev so pokazale, da brezpilotno letalo ne sme biti ponovno vklopljeno, saj ponovni prižig letala spremeni podatke z ustvarjanjem nove datoteke DAT in lahko izbriše shranjene podatke, če je notranji pomnilnik brezpilotnega letala poln.

Programming, investigation and documentation in digital investigation

Ahmed Hisham Radwan
ah4965@student.uni-lj.si
FRI

ABSTRACT

This paper discuss the amount of effort and time required to hold a Digital investigation in terms of programming, investigation also the documentation and logging of what is done. The paper follow the experiment discussed in the paper "Do digital investigators have to program? A controlled experiment in digital investigation". The experiment research a lot of aspects regarding the process of a investigating a digital involved crime, those aspects differ from the relation between amount of effort in terms of time invested in the investigation with the quality of the results, to the need of programming skills or the implementation of any programs through out the carrying on of a digital investigation. The experiment was held on 39 students who were split in 10 groups all of are computer science graduate students.

Keywords

Digital forensics, programming, investigation, documentation

1. INTRODUCTION

The role of programming, investigation and documentation, and the amount of time invested in each of those parts of a digital investigation is a very important matter in terms of improving the digital investigation and understanding the process it go through. Also the amount of experience and knowledge of programming and different aspects of computer science needed by the investigator and whether or not a strong computer science background is required continue to be a debatable topic. In the research conducted and published in the paper "Do digital investigators have to program? A controlled experiment in digital investigation" by Felix Freiling and Christian Zoubek, some solid results are shown that can expand more our understanding of the role and amount of time invested in programming, investigation and documentation plus a little shadow on the of experience and knowledge of programming and different aspects of computer science.
The experiment that is going to be discussed was conducted at Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) in Erlangen, Germany.

2. THE EXPERIMENT[1]

The research is based on a controlled experiment. The experiment used an approach called case-based reasoning (CBR) which is commonly used to study general work of investigations and investigators by extracting knowledge from prior cases to be used in the future to solve new ones. All the participants of the experiment are graduate students of Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) in Erlangen, Germany. The number of those participants are 39. The participants were divided into 10 groups.The groups were set to solve one of three realistically claimed cases. The time frame in which the groups were expected to finish the experiment and submit their final findings was set to eleven months.

The main purpose of the research was to analyze the participants throughout the experiment. The research didn't have a hypothesis but rather exact questions that was expecting thought the analyzing of the work of the participants to answer. The questions were about the effect of effort in terms of time on the quality of the investigation result, the previous grade of the participants relation to their results quality, how the work done is divided between investigation, documentation and analyzing and is it necessary to program to solve the case.

2.1 Setting up the research

The research is set about simulating a real life like case. The case consists of context of the investigation and investigation goals and also digital evidence that require analyzing. The effort done in the investigations is measured in the amount of minutes of work spent in solving the cases. In the experiment there is four different type of work :

- Documentation and work involving pen and paper.
- Group discussions and meet-ups.
- Programming or using old ones with new implementation or add ups.
- Using tools to actually preform a digital investigation on the evidence in hand.

The quality of the findings were evaluated as a percentage of important evidence found and how correctly interpreted by the participants.

At the beginning the participants were introduced to the nature of the experiment. Participating in the experiment was mandatory, however it didn't have an effect on the grades of the participants. After the participants were divided into groups they were instructed not to interact in any way concerning the experiment especially the groups that share the same case. As a procedure to beginning with the experiment the participants were asked to fill out a questionnaire where they had to give out information about there grades, degree, year of study, experience in working on forensic cases, motivation and expected effort to invest in the experiment.

Based on the lack of knowledge about what influence the digital investigation process. The experiment was designed with exploratory research questions rather than try to prove a raised hypothesis. Those question were raised based on four criterion :

- The characterization of the difference in case types.
- The observation of investigative strategies.
- What influence the overall effort ?
- The factors influencing the quality of result.

In the characterization of the difference in case types criteria, the question raised were if the known rule that every case must be treated with an open mind so not to pass some important relevant evidence always stand, or there is some type of cases that would always require more effort. With that in mind the following to questions were asked :

- Is there a difference in the effort invested by every group to solve different cases?
- Is there a difference in the effort invested by every group to solve the same case?

When observing the investigative strategies, its interesting how each group attempted to solve the case and the ordering of the steps taken to solve the case, thus those questions were generated:

- Did group with different cases used different strategies in attempt to solve the case?
- Is the amount of effort distributed differ between different groups and different cases?
- What is the tools used in the attempt to solve the cases?

In the attempt to answer the question of what influence the overall effort invested to attempt to solve the cases given to the groups of participants in the experiment so that it can be know how different factor; for example programming skills,experience motivation, etc influence total effort. To answer that two more question expanded from the original one:

- What are the main factors that relate to the total effort in every case?
- What exact factor can give good prediction of the effort?

The last question were related to the results. As known one of the main goals of an investigation is to produce high end quality results. So the factors influencing the quality of result is a very important topic that required to take part in the setting of the experiment. The two questions regarding the factors influencing the quality of result where :

- What are the main factors that relate to the quality of result?
- What exact factor can give good prediction of the result quality?

2.2 The cases

Every group was assigned one of three cases. Those cases simulate real life like cases. The also served in a way to show how different cases can effect effort. The three cases were created from a total of four cases, that were inspired by real life cases reported to the police force of of the federal state of Baden-Württemberg, Germany. Due to lack of experience in developing cases, one case was used in an earlier forensic course to as an analysis exercise. The experience gained from using one case as an analysis exercise helped making the three cases used in the experiment less artificial and more real life like.That ended into three cases which are pointed to with the names : The ARPspoof case,the Terror case and the Malware case. The evidence used for the cases were disk images.Every case had at least three disk images. The disk images used as evidence were designed so not to be analyzed simultaneously but to be analyzed after the other. This was done by making every disk image having evidence pointing towards the next disk images.
The experiment cases were designed as event based, were every event unveil the next event. There were four major meetings that were made to be were new events in the case happens. The meetings were made with the course instructor and he was assumed as a role play were the course instructor is the prosecutor and the students the investigators. The main role of the meetings is to review the progress made by the student were they discuss their finding and also the advance of the case where the course instructor acting as the prosecutor will suggest the acquiring of more evidence. The evidence will be the next image disk and the experiment was designed in a way were most groups have equal number of disks in the same time. At their first meeting the groups are instructed about the case by the course instructor and handed the first disk image. The second and the third meeting purpose was review the progress made by the student were they discuss their finding and also the advance of the case where the course instructor acting as the prosecutor will hand on the next two disk images respectively. The last meeting wasn't part of the experiment or the cases but more of a feedback on the groups report and the whole experiment.

2.2.1 The ARPspoof case

The ARPspoof case is a case were an system administrator of corporate is suspected of spying on one of the network users. The suspicion was raised due to the administrator

was noticed to talk about knowing log-in information about the victim publicly. The administrator preformed a man in the middle attack were he directed the data frames sent by the victim to his computer rather than sending them to the gateway and also changed his internet traffic by redirecting him to other web pages when try to visit specific pages.
The image disk for this cases were as following:

- The administrator computer.
- The victim computer.
- The gateway router.

In the end report a grade was given based on finding those key findings:

- Comparison of time stamps on the three image disks.
- Comparison on the acquired data and the victim actions.
- Checking the http IDs.
- Checking the data from the DNS protocol direction.

2.2.2 The Terror case

The terror case as can be guessed by the name is a terrorist bombing attack based case. This case due to sensitivity that may rise followed an episode of star trek were the embassy of the united earth was bombed on the planet Vulkan. The suspect and a used coordinated the attack by using some communication forum to hide their real communication were they spammed with virtual users who uses random bible versus to submit topics. The communication between the two took place in between that versus spamming. The image disk for this cases were as following :

- The suspect computer.
- The server where the forum ran.
- The user who the suspect communicated with computer.

In the end report a grade was given based on finding those key findings:

- The finding of sessions IDs ,IPs and cookies.
- Comparison of time stamps on the three image disks.
- Prove the link between the computers using log files.
- The login data to the server of the suspect and the user.
- The randomly spammed bible versus script.
- The communication history in both the suspect and the user browsers .

2.2.3 The Malware case

The malware case handle a case were a blog server is infected with a malware. The infection caused the spread of the malware by getting users infected too. After the infection the malware collected data from the infected user and send them to a drop-zone were all the data were collected. The infection was due to vulnerability in the PHP code of the website. The first infection was the owner of one of the blogs and the second victim was just visiting the blog. After the infection begin the malware spread to the blogs that were hosted on that server. The image disk for this cases were as following :

- The blog's owner computer that was initially infected and the server that was suspected to be the root of the infection.
- If identified the attacker computer was handed, in addition to another victim computer.
- If a connection to the server was proven an imgae representing the drop-zone was handed.

In the end report a grade was given based on finding those key findings:

- The browser history and IP addresses.
- Comparison of time stamps on the disk images.
- Prove the link between the computers using log files.
- Comparison of the malware through out the computers.

3. THE RESULTS

3.1 Demographic data of participants

Let us first give an overview of the participants from the pre-questionnaire. Overall, there were 39 participants that got randomly assigned to 10 groups with one group consisting of only 3 participants. That group did not hand in project diaries so they were dropped from the evaluation. Two more participants also did not hand in the project diaries so therefore project diary/effort data was collected from 34 participants in total.

3.2 Case types and investigative strategies

In order to understand if groups used different strategies to solve different cases, effort was plotted on a timeline that covered the entire study period. Unlike for Malware and Terror cases, where effort is clustered in three chunks around the dates when groups met with the prosecutor, the effort put in ARPspoofing seems more evenly distributed as well as less in total. In the beginning all groups would focus more on technical analysis and then shift to more documentation, which is not unexpected. Since in practice the results must stand in court, investigators might choose a more quality-oriented approach.
When total effort per case is considered, we can see that the Terror case required by far the most effort. The other two are somewhat close in effort required to finish the task. This could be due to the Terror case being less clearly specified than the ARPspoof and Malware cases. The Terror cases also required analysis of large texts and discovering of motives.

If we take a look at the amount of effort invested in different tasks, technical analysis is dominating the spectrum and is followed by conceptual work and group meetings, while no group spent time on programming which means that existing tools were enough to solve the cases. It is interesting that the only tools that were used are the ones that are available to the public meaning no commercial forensics software was needed. The tools used were:

- Fred
- Sleuthkit and Autopsy
- Certutil
- Virtual Box
- Last activity view
- IDA Pro (free version)
- OllyDBG
- DBBrowser

One explanation as to why the participants did not program is that these tools are sufficient enough to solve the cases. The other explanation could be that the effort to program new tools/scripts in order to automate investigative steps does not pay off for an individual case. The unfortunate thing here is that since the participants did not program, the question about the role of programming knowledge is left unanswered.

3.3 Factors influencing effort

Next, individual participants' data is taken and the factors that correlate with total effort per case are investigated. Planned effort and motivation correlate, which is expected, but planned effort does not correlate with actual/real effort since real effort was usually lower than planned effort. This can probably be attributed to the fact that participants were still relatively inexperienced. Real effort also does not correlate with motivation which is surprising. Furthermore, higher motivation usually resulted in lower real effort which might be explained by higher efficiency in solving the cases.

3.4 Factors influencing quality

With efficiency being an aspect of result quality, the relation between group motivation and result quality was investigated. While total effort positively correlating with quality of result was not surprising, what was interesting is the average motivation of the group members inversely correlates with the result quality. So, we can safely say that subjective motivation does not appear to be a good predictive factor for quality.

When it comes to the correlation between the grade of the participants who participated in the basic course of digital forensics before and the result quality in this study, some methodological problems occurred. Result quality, or grade, is a group attribute, and the grade in the basic digital forensics course is individual attribute that only a part of all participants have. To make the problem even harder, participants weren't asked to give their exact grade but rather a range in which their grade belongs. This was solved by using the group grade as individual grade and only taking into account those participants that participated in the basic course. Unsurprisingly, previous grades seem to be a good predicting factor for future grades since there's a positive correlation between these two things.

4. CONCLUSIONS

While this experiment may not be statistically significant, it could be used to help further research in this area. Questions about correlating factors, such as total effort, motivation and result quality, may be interesting for further research. In future, the difference between individual and group motivation, effort and grade should be clearly made so as to not run into the problem explained in the results part. It's recommended that, in order to get more accurate and reliable results, a longer study should be done. The study should include more factors taken into account as well as participants working individually on several cases instead of groups of participants working on single case.

As far as case comparison is concerned, different parameters concerning the case, e.g., the complexity and number of files to interpret, number of disk images to be analyzed, should be more precisely assessed. Notes were also taken by the instructors but were unusable since they were not structured in any particular way. While the cases used in this study may not be a great example of the common cases that investigators encounter in real practice, it might be interesting to look into the effect of different tools and knowledge about programming on the results, since there is a dispute in practice about what level of programming knowledge should be required.

It is also recommended that the experience of the participants be taken into the consideration in the future. A study involving a 100 experienced digital investigators in a controlled environment should obtain results that can be statistically useful. However, this might be unrealistic considering the shortage of resources and personnel required for such a study.

5. REFERENCES

[1] F. Freiling and C. Zoubek. Do digital investigators have to program? a controlled experiment in digital investigation. *Digital Investigation*, 20:S37–S46, 2017.

SCARF: Skaliranje digitalnega forenzičnega procesiranja z uporabo vsebniških tehnologij v oblaku

[Razširjen povzetek]

Janez Eržen
Fakulteta za računalništvo in informatiko
Večna pot 113
Ljubljana, Slovenija
je1468@student.uni-lj.si

Uroš Bajc
Fakulteta za računalništvo in informatiko
Večna pot 113
Ljubljana, Slovenija
ub6189@student.uni-lj.si

ABSTRACT

V zadnjih letih je zmogljivost računalnikov močno zrasla. Posledično so se povečale tudi kapacitete podatkovnih nosilcev oz. diskov. Temu razvoju bodo morali slediti tudi forenzični postopki, da se bodo lahko prilagodili na zmogljivejše računalniške vire.

S tem namenom so bile razvite nove programske rešitve, ki omogočajo odprto in fleksibilno integracijo že obstoječih orodij, poleg tega pa podpirajo skalabilnost in prilagodljivost na zahtevnejše procesiranje podatkov. Eno od tovrstnih ogrodij je ogrodje SCARF [13]. Uporablja princip vsebnikov (angl. *containerization*) ter s tem omogoča vzporedno procesiranje podatkov, ki se skalira glede na zahtevnost dane naloge. Poleg tega omogoča tudi enostavno dodajanje razširitev.

Ključne besede

digitalna forenzika, vsebniki, oblačna forenzika

1. UVOD

Forenzično preiskovanje digitalnih dokazov v dodeljenih rokih postaja zaradi vedno večjih diskov velik izziv. V enem od laboratorijev za forenzične raziskave (RCFL) pod okriljem FBI so bile izvedene raziskave, ki so pokazale, da je kar 60% raziskav trajalo dlje od 90 dni, skupno 16.7% pa jih je bilo odprtih kar več kot leto dni [14].

V splošnem je tovrstne raziskave znotraj predpisanih rokov težko izpeljati zaradi nekaj glavnih razlogov: vedno več je podatkov za procesiranje in preiskovanje, zahtevnost preiskovanja je vedno večja; kadrovsko se laboratoriji ne širijo sorazmerno z zahtevami preiskovanja. V ta namen je potrebno zagotoviti in nadgraditi računalniško podporo preiskovanju, da bo mogoče naloge opraviti v želenem času.

Prva pomembna stvar je zagotavljanje dobre strojne opreme. Forenzični laboratoriji bi morali izkoriščati prednosti novih SSD diskov, ki omogočajo hitrosti višje od 1 GB/s v primerjavi s počasnejšimi HDD različicami. Druga pomembna stvar je uporaba hitre omrežne tehnologije, kot na primer uporaba hitrih omrežnih stikal (*Infiniband*), kar bi spremenilo dejstvo, da omrežje in vhodno izhodne operacije predstavljajo ozko grlo računalniškega procesiranja podatkov. Glavna težava pa ostajajo procesorski viri.

V ta namen so se pojavile ideje po izdelavi odprte, celovite in skalabilne platforme, ki bi ponujala ključne storitve ter preprosto ter posledično poceni integracijo z obstoječimi rešitvami. Odprta pa predvsem zaradi tega, ker so tovrstne rešitve bolj uporabljene in testirane, napake se hitreje zazna in odpravi, kar prinaša večje zaupanje uporabnikov.

Nekaj tovrstnih rešitev že obstaja, a večinoma ne omogočajo preproste integracije. Najbolj popularno izmed tovrstnih orodij je orodje TSK [6], ki pa ni načrtovano za uporabo v skalabilnih okoljih. Obstajajo že neuspešni poizkusi uporabe in razširitve dela orodja TSK - *Sleuthkit* v ogrodju za obdelavo masovnih podatkov (angl. *Big Data*), ki pa so hitro propadli.

Rešitev bi morala temeljiti na preglednih programskih vmesnikih, neodvisnih od programskega jezika, ter uporabljati skupno bazo dokazov. V ta namen je bila zamisel zasnovana na lahki virtualizaciji znotraj med seboj neodvisnih vsebnikov, ki ponuja preprosto razširljivost in prilagajanje obremenitvam.

2. PODOBNE REŠITVE

Zaradi potreb po hitrem preiskovanju forenzičnega gradiva je bilo v zadnjih letih razvitih kar nekaj rešitev podobnih ogrodju SCARF.

Večina teh sistemov ne ponuja celotne rešitve, oz. se osredotočajo na ožja ali druga področja. Eden od takih pomembnih projektov je projekt *Hansken*, razvit na nizozemskem forenzičnem inštitutu [4], ki se osredotoča na procesiranje masovnih podatkov (angl. *Big data*). Njegove glavne pomankljivosti so, da ne gre za odprtokodno orodje in da ni možno korektno izmeriti učinkovitosti tega izdelka.

Druga od rešitev na področju digitalne forenzike, ki jo omenja članek [13], je orodje za masovno ekstrakcijo (angl. *bulk extraction*) [8], ki omogoča procesiranje datotek in slik v velikih količinah. Uporablja prevedene regularne izraze, ki jih nato izvaja večnitno v do 48 jedrih. Glavni pomanjkljivosti tega orodja sta slabi podpori za obdelavo datotek in integracijo v stroke. (angl. *clusters*).

Eno od uporabnih orodij je tudi *ExifTool* [2], namenjen za procesiranje metapodatkov iz slik. Ima dobro podporo za integracijo, tako da ga je preprosto vključiti v nove rešitve kot modul.

Osrednja tema članka [11] je zelo podobna, in sicer gre za procesiranje forenzičnega gradiva v oblačnih, predvsem IaaS arhitekturah. Hkrati se članek ukvarja tudi s težavami, ki se pojavljajo z integriteto, zasebnostjo in dostopnostjo znotraj tovrstnih okolij, na katere se pričujoči članek ne osredotoča preveč.

Tudi članek [12] se ukvarja s tovrstnimi problemi. Predvsem se osredotoča na zmogljivosti takega ogrodja. V idealnem primeru bi se morala zahtevnost procesiranja podatkov in s tem poraba računalniških virov večati sorazmerno s količino podatkov za obdelavo.

V kratkem, obstaja kar nekaj raziskav narejenih na področju digitalne forenzike, ki se ukvarjajo z učinkovitim preiskovanjem digitalnih dokazov z uporabo računalniške tehnologije, ki pa večinoma niso celovite in v večini primerov niso odprtokodne, kar pomeni, da so premalo uporabljane. Iz tega sledi, da se takih rešitev ne dopolnjuje in razširja, da bi omogočale celovito pokritost omenjenega področja. Z ogrodjem SCARF so avtorji poskušali odpraviti prav to pomanjkljivost.

3. VSEBNIKI

Vsebniki trenutno zelo hitro pridobivajo na priljubljenosti. Njihovi začetki izvirajo iz razvoja mehanizmov za omejevanje dostopa do drugih datotek. Prvi tak sistem imenujemo 4.2BSD [10], njegove glavne naloge pa so bile razvoj in testiranje novih funkcionalnosti na Unix operacijskih sistemih. Kasneje se je razvil v bolj zaprt in dodelan sistem vsebnosti, ki je omogočal dve glavni funkcionalnosti: poganjanje določene aplikacije z omejenimi pravicami ali pa kreiranje navideznih slik, ki so poganjale različne storitve in procese v ozadju. Eden od pomembnih projektov, ki je pospešil razvoj vsebnikov, je bil projekt VServer [7]. Zasnovan je bil na osnovi sistema Linux in je omogočal delovanje več instanc Linux serverja na eni napravi, ki so tekle samostojno in zavarovano. Ta koncept so uporabili ponudniki internetnih storitev, ki so z uporabo tovrstne tehnologije omogočili gostovanje na virtualnih zasebnih strežnikih.

Leta 2007 je bil predstavljen pojem "splošno namenskih vsebnikov". Do neke mere so bili podprti z uveljavljanjem kontrolnih skupin (angl. *control groups*) kot del verzije 2.6.24 sistema Linux. V kasnejši verziji 3.8 so bili dodani še uporabniški domenski prostori (angl. *user namespaces*), ki so dovoljevali boljše omejevanje dostopa do priklopljenih nosilcev, drugih procesov ter omrežja. Na tej osnovi so bile kasneje razvite številne rešitve: LXC je ena izmed prvih, najbrž najbolj znana je Docker, poleg dobro uveljavljenega Googlovega izdelka Kubernetes, ter ostalih, RKT ter LXD.

Zaradi hitrega razvoja tovrstnih ogrodij je bilo ustanovljeno združenje imenovano *Open container inovative*, ki je specificiralo formate slik diskov ter vmesnike za rokovanje s procesi, ki tečejo na teh slikah. Ti formati naj bi omogočali interoperabilnost in konkurenco med razvijalci tovrstnih ogrodij.

3.1 Docker in vsebniki

Za ogrodje SCARF so avtorji članka [13] izbrali Docker kot orodje za orkestracijo vsebnikov. Le-to omogoča zelo dobro in poglobljeno manipulacijo vsebnikov. Njihovo kreiranje, posodabljanje, verzioniranje, kreiranje večih instanc istega vsebnika, dodeljevanje virov, zaustavitev itn. Vsi vsebniki si delijo isti operacijski sistem, a se med sabo običajno ne vidijo, razen če to posebej nastavimo. Vsebniki se obnašajo zelo podobno kot virtualne naprave (angl. *Virtual machines*), le da porabijo veliko manj računalniških virov za kreiranje in zaustavitev v primerjavi z "virtualkami".

V vsakem od vsebnikov teče en ali več procesov, ki imajo dodeljenih nekaj računalniških virov: procesorskih jeder, začasnega pomnilnika (RAM), datotečnega sistema, omrežno povezavo itn. V primerjavi s procesi vsebniki omogočajo večjo avtonomijo, boljše upravljanje in prilagajanje. So samostojne enote, ki vsebujejo vse potrebno za svoje delovanje: od programske kode, podatkov, pa do konfiguracije okolja.

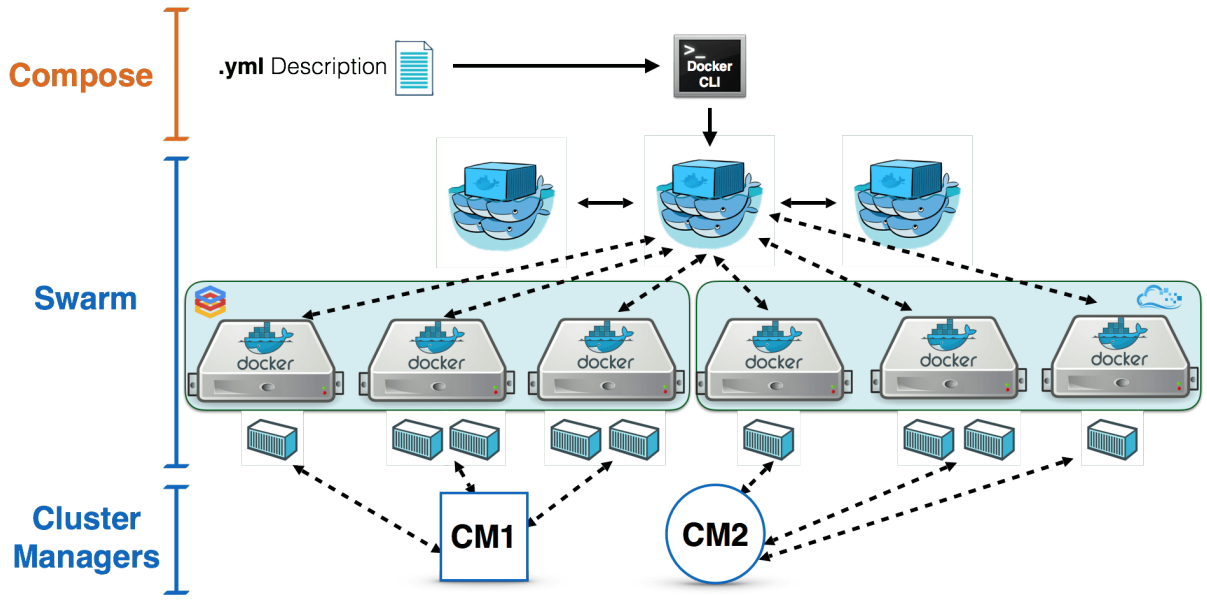
3.2 Upravljanje z vsebniki

Za uporabo vsebnikov je potrebna tudi uporaba katere od platform za orkestracijo z njimi. Tovrstne platforme skrbijo za upravljanje z vsebniki, dodeljevanjem potrebnih virov, ustvarjanje večih instanc vsebnika, ko je potrebno in podobno. Avtorji članka [13] so se odločili, da bodo za ta namen uporabili *Docker Swarm*.

Swarm sestoji iz množice *vozlišč* (angl. *Docker engines or nodes*), organizirane v stroke (angl. *clusters*), na katere so nato nameščene raznolike *storitve* [9]. Vse upravljanje z vsebniki, instancami, storitvami ter ostalim je možno preko preglednega vmesnika (API-ja) oz. iz terminala (CLI). Za upravljanje z instancami skrbijo *upravljalska vozlišča* (angl. *manager nodes*), ki skrbijo za izvajanje nalog, ki jih izvršujejo delavna vozlišča (angl. *worker nodes*). Le-ta sprejmejo nalogo in jo izvršijo po navodilih upravljalskih vozlišč, ki skrbijo za ravnovesje in skalabilnost sistema. Eno od upravljalskih vozlišč je izvoljeno kot vodja, ki skrbi za koordinacijo med preostalimi vozlišči. Osnovna shema je predstavljena na sliki 1.

Glavna enota, katero definira razvijalec na platformi Swarm, je storitev. Ta nosi podatke o tem, katera slika bo pognana znotraj določene naloge (angl. *task*), s kakšno konfiguracijo, začetnimi računalniškimi viri, ter na kakšen način in do kakšne mere se bo ta slika replicirala. Vsaki od storitev so ob inicializaciji dodeljena določena omrežna vrata, preko katerih lahko do nje dostopamo. Upravljalska vozlišča skrbijo za porazdeljevanje zahtev med posamezne instance storitev, ter pri prevelikih obremenitvah za njihovo repliciranje. Na ta način je omogočeno hitro in prilagodljivo okolje, ki precej optimalno upravlja z računalniškimi viri.

Tak pristop je ključnega pomena za učinkovito procesiranje



Slika 1: Shema Docker Swarm arhitekture[5]

digitalnega forenzičnega gradiva, zato so se ga poslužili tudi avtorji sistema SCARF.

4. SCARF

V preučenem članku [13] je opisan prototip rešitve za procesiranje forenzičnih podatkov, ki so ga avtorji članka razvili in temelji na vsebnikih Docker. Imenuje se SCARF (SCAlable Realtime Forensics), kar v prevodu pomeni skalabilna forenzika v realnem času. Na prototipu so lahko preizkusili, kako se tak pristop obnese v praksi in definirali njegove prednosti in slabosti.

V prototipu se uporabljajo vsebniške tehnologije, ki zapakirajo individualne izvršljive module v avtonomne slike, ki za delovanje potrebujejo le osnovno namestitev operacijskega sistema. Instance teh slik so definirane kot *opravila*, ki predstavljajo osnovne enote dela, ki ga opravlja procesor. Opravila se lahko izvajajo kot posamezni procesi ali pa kot skupine procesov, ki opravljajo določeno funkcijo ter delujejo v zaprtem okolju.

Pričakovati je, da bo uporaba vsebnikov omogočila učinkovito skaliranje operacij, ki se lahko izvajajo vzporedno (to so operacije, ki se izvajajo na nivoju posameznih datotek, npr. računanje zgoščenih vrednosti datotek). Prav tako naj bi bilo razširjanje z novimi moduli, ki so namenjeni drugačnim funkcionalnostim, cenovno ugodno. Platforma sestoji iz komponent, ki so opisane v naslednjih poglavjih.

4.1 Vnos podatkov

Za potrebe testiranja so bile uporabljene slike NTFS, ki so bile pred dejansko uporabo pred-procesirane; najprej se je namreč prebralo glave NTFS, ti podatki pa so bili potem tekom procesa shranjeni v pomnilniku. S shranjenimi metapodatki se lahko optimizira pridobivanje posameznih podatkov z diska. V primeru da so bloki, ki vsebujejo datoteke, fizično postavljeni v bližini, se lahko podatke bere linearno, če pa so ti fragmentirani po disku, je branje bolj težavno. Zato so podatke raje najprej prebrali v RAM (v kolikor so bili dovolj majhni) in jih šele potem posredovali dalje.

4.2 Arhitektura

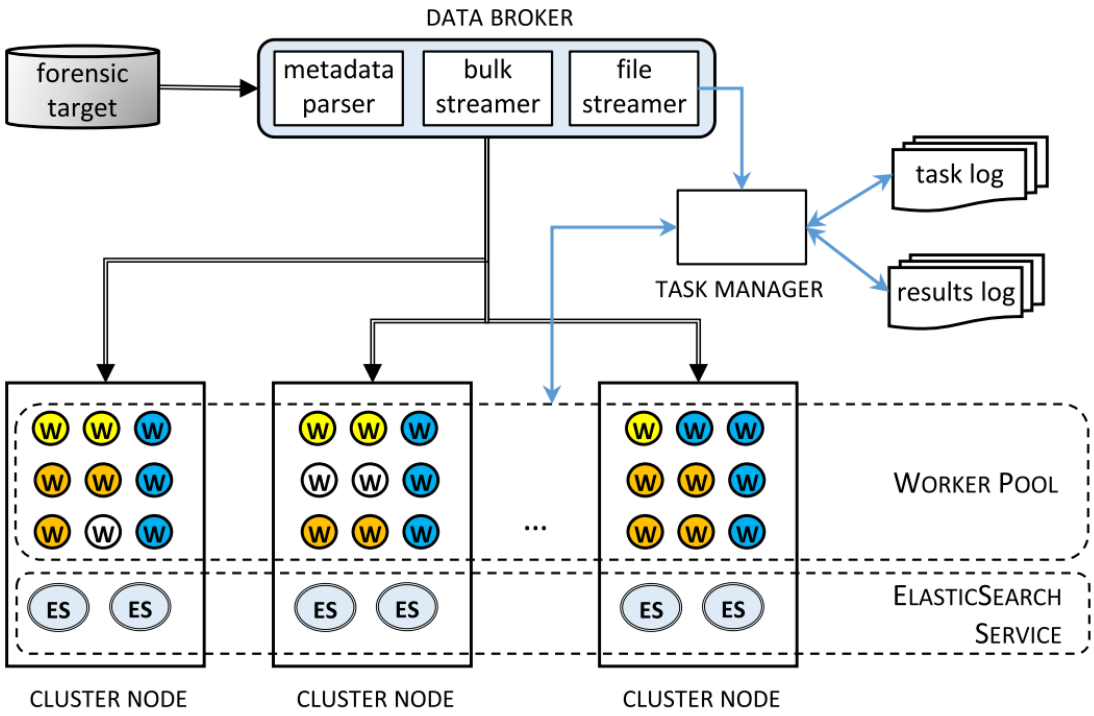
Na sliki 2 so prikazane glavne komponente platforme, vidi se tudi, kako poteka izmenjava podatkov med posameznimi komponentami. T.i. *posrednik podatkov* (angl. *data broker*) je zadolžen za pridobitev podatkov iz *forenzične tarče* (angl. *forensic target*) ter njihovo pripravo. Podatke potem posreduje posameznim vozliščem v gručah. Posrednik podatkov je tako nekakšen abstrakten nivo, ki loči procesiranje podatkov od njihove predstavitve.

4.2.1 *Upravitelj opravil (angl. Task manager)*

Ta komponenta skrbi za shranjevanje zabeležk o definicijah opravil, ki jih definira posrednik podatkov, in uspešnosti njihovega opravljanja. Za vzdrževanje obstojnih zabeležk in obveščanje registriranih podatkovnih klientov (angl. *data client*) upravitelj opravil uporablja *Apache Kafko*. Zanesljivo shranjevanje zabeležk je zelo pomembno, saj v kompleksnih distribuiranih sistemih dokaj redno prihaja do napak, ki jih je težko preprečiti. Pri nekaterih od mnogo instanc vsebnikov lahko namreč zaradi različnih dejavnikov pride do napake pri zagonu. V kolikor ponovni zagon instance ne reši težave, pridejo prav natančne zabeležke, s katerimi si je mogoče pomagati pri razhroščevanju.

4.2.2 *Odjemalci podatkov (angl. Data clients)*

Odjemalci podatkov so vsebniške instance, ki vsaka posebej sprejmejo delež forenzične tarče in na njej organizirajo izvajanje posameznih opravil. Samega izvajanja opravil pa ne



Slika 2: Arhitektura platforme SCARF

izvajajo odjemalci, ampak so ta posredovana naprej v dinamičen bazen posebnih vsebnikov, ki se imenujejo *delavci* (angl. *workers*).

4.2.3 Delavci (angl. *workers*)

Delavci so instance vsebnikov, ki opravljajo specifična opravila na delih podatkov, ki so jim dani. Delavci izpostavljajo vmesnike za procedure preko oddaljenjega dostopa (vmesnike za RPC), kjer sprejemajo podatke in nato vračajo rezultate v obliki niza JSON.

Uporaba vsebnikov omogoča delavcem, da se enostavno skalirajo, glede na količino dela, ki ga morajo opraviti. Ta način tudi omogoča razvrščanje opravil po prioriteti, kar pomeni, da se lahko razvije metode, ki avtomatično skalirajo vsebnike, glede na porabo virov na strežniku ali pa pomembnost opravil. Docker to omogoča z uporabo ukaza *docker service scale*.

Odjemalci podatkov nimajo informacij o tem, koliko delavcev je na voljo, niti na katerih mrežnih vratih so dostopni. Za dostop in za osnovno porazdeljevanje obremenitve skrbi notranja storitev DNS.

4.2.4 Repozitorij rezultatov (angl. *results repository*): ElasticSearch

Ko se opravila zaključijo, se njihovi rezultati vrnejo odjemalcem podatkov. Serije več rezultatov se nato pošljejo gruči ElasticSearch (ES) vozlišč. Na teh vozliščih so tako podatkovne baze z rezultati, do katerih se lahko dostopa preko vmesnikov REST, ki omogočajo shranjevanje, iskanje in vračanje podatkov. Tudi ti se lahko v primeru povečanega števila zahtevkov dinamično skalirajo.

4.2.5 Upravljanje z vsebniki

Za upravljanje z vsebniki skrbi storitev, ki teče v ozadju na vseh fizičnih vozliščih. V kolikor imajo obstoječi vsebniki preveč dela, storitev poskrbi za dodajanje novih vsebnikov in nato za njihovo odstranjevanje, ko se obseg dela zmanjša. Obenem tudi nadzira njihovo delovanje in v primeru napak poskrbi za njihov ponoven zagon. Nudi tudi sistem za odkrivanje storitev.

4.2.6 Razširitve platforme SCARF

Poleg enostavnega in učinkovitega skaliranja je bil cilj prototipa SCARF omogočiti enostavno dodajanje novih tipov delavcev. Ker delavci v osnovi "le" sprejmejo podatke, opravijo svoje delo, in vrnejo niz JSON, je dodajanje novih tipov delavcev v vsebnike zelo enostavno. Seveda morajo tudi ti znati razumeti podatke, ki jih prejmejo in vrniti niz ustrezne oblike. Tako dodajanje novih orodij v SCARF ponavadi sestoji iz treh korakov: dodajanja ovojnice RPC, vstavljanja orodja v vsebnik in zaganjanja orodja.

Namen ovojnice RPC je, da izpostavlja vmesnik, preko katerega se pridobijo podatki, in mehanizem, ki skrbi za vračanje rezultatov. Tako skrbi za to, da se orodje izvede nad prejetimi podatki in vrne rezultate SCARF-u. Samo orodje mora biti skupaj z ovojnico vstavljeno v sliko vsebnika. To se pri vsebnikih Docker naredi z uporabo datoteke *Dockerfile*. V

njej se sprva definira začetna slika operacijskega sistema, na katerem bo orodje v tem vsebniku teklo. Dalje sledijo koraki kot so namestitve različne dodatne programske opreme, definicije spremenljivk okolja, itd. Določena je tudi namestitev ovojnice, ki se izvede ob stvaritvi vsebnika. Ko je orodje vstavljeno v vsebnik in izpostavljeno kot mrežna storitev, je pripravljeno za delovanje.

5. ANALIZA UČINKOVITOSTI DELOVANJA

Ker je namen platforme SCARF skaliranje operacij forenzičnega procesiranja v primeru povečanega obsega dela, so bila za testiranje skaliranja uporabljena različna orodja za forenzično rabo. Cilj je bil pokazati, da večje število vsebnikov rezultira v povečanem pretoku obdelave podatkov.

Za testiranje je bila uporabljena gruča štirih strežnikov povezanih z 10 GbE stikalom. Vsak strežnik ima na voljo 256 GB RAM-a, 24 2.6 GHz dvonitnih jeder, kar skupaj pomeni 96 fizičnih jeder, oz. 192 logičnih jeder. Vsako vozlišče ima 1 TB SSD disk. Pretok preko TCP povezave za masovni prenos (angl. *bulk transmission*) je bil približno 1 GB/s. Testiranje je potekalo nad 200 GB veliko NTFS sliko, ki je bila ustvarjena z uporabo naključnega izbiranja med datotekami t.i. GovDocs korpusa [8]. Za potrebe testiranja in zaradi lažje analize, je bil vsakemu vsebniku dodeljeno le eno CPU jedro.

5.1 Šifrirno zgoščevanje

V forenziki se zelo pogosto uporablja *šifrirno zgoščevanje*. V tem primeru je bil izbran SHA1 algoritem, pri čemer je bil ustvarjen vsebnik, ki nudi oddaljen klic preko TCP povezave in vrne zgoščeno vrednost danih podatkov. Rezultati so prikazani v tabeli 1, videti je, da že 12 vsebnikov skoraj v celoti zapolni pasovno širino, ki je na voljo.

Št. vsebnikov	4	12	24	48	96	192
MB/s	345	857	985	985	948	992

Tabela 1: Pretok SHA1 zgoščevanja v odvisnosti od števila vsebnikov

5.2 Ekstrahiranje metapodatkov

Orodje *ExifTool* [2] se uporablja za pridobivanje metapodatkov iz datotek. Rezultati testiranja so prikazani v tabeli 2. Vidno je skoraj linearno povečanje pretoka (s 5 MB/s pri 4 vsebnikih do 192 MB/s pri 192 vsebnikih). Rezultati so v skladu s pričakovanji, saj pridobivanje metapodatkov ni odvisno od vhodno/izhodnih operacij, prav tako je možen visok paralelizem. Upoštevaje celoten razpon parametrov testiranja se lahko vidi, da povprečen pretok na vsebnik sledi binomski porazdelitvi z najvišjo točko pri 32 vsebnikih.

5.3 Klasifikacija slik

Testirana je bila tudi uporaba odprtokodnega orodja za klasifikacijo slik OpenNSFW [3]. Ta globoka nevronska mreža je naučena detekcije pornografskih slik. Preizkušanje obnašanja tega orodja na SCARF-u prinaša zanimive rezultate, saj so operacije, ki se izvajajo pri klasifikaciji, zelo kompleksne in potratne. Kot se lahko vidi v tabeli 3 se število

Št. vsebnikov	4	8	24	48	96	192
MB/s	5.2	17	99	151	170	192
MB/s na vseb.	1.3	2.1	3.1	2.4	1.8	1.0

Tabela 2: Pretok ekstrahiranih podatkov z uporabo ExifTool v odvisnosti od števila vsebnikov

klasificiranih datotek na sekundo povečuje skoraj linearno, je pa pretok v MB/s v primerjavimi z ostalimi orodji precej manjši. Kljub temu je število klasificiranih slik na uro mnogo večje, kot če bi bilo to potrebno početi na roke. Je pa v nasprotju s prej testiranimi orodji OpenNSFW že zapakiran v docker vsebnik, tako da je integracija v SCARF še lažja in hitrejša.

Št. vsebnikov	4	8	12	24	48	96	192
MB/s	0.4	1.4	2.5	3.8	7.2	10.9	21.3
Datotek/s	0.8	2.2	3.9	7.1	13.4	20.3	38.5

Tabela 3: Pretok OpenNSFW klasifikacije slik v odvisnosti od števila vsebnikov

5.4 Indeksiranje pogostih tipov datotek

Pridobivanje čistopisa iz zakodiranih datotek se v forenziki izvaja zelo pogosto. Preizkušeno je bilo odprtokodno orodje *Apache Tika* [1]. Rezultati 4 kažejo, da je ekstrahiranje zelo zahtevno, saj pride le do sublinearnega skaliranja. Še posebej opazen je padec učinkovitosti pri 192 vsebnikih, kar se lahko pojasni v tem, da je na voljo le 96 fizičnih jeder in da Apache Tika dobro izkorišča vse enote CPU-ja ter tako uporaba večjega števila niti ne nudi izboljšav.

Št. vsebnikov	4	12	24	48	96	192
MB/s	0.5	1.1	2.4	3.5	5.8	6.7
MB/s na vseb.	0.13	0.09	0.10	0.07	0.06	0.03

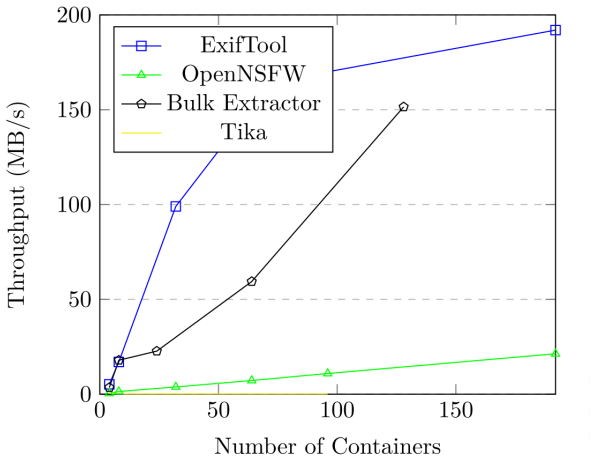
Tabela 4: Ekstrahiranje čistopisa z uporabo orodja *Tika* v odvisnosti od števila vsebnikov

5.5 Masovna ekstrakcija (angl. *bulk extraction*)

Bulk Extractor [8] je orodje za analizo surovih podatkovnih tokov. Z uporabo pred-prevedenih skenerjev, ki bazirajo na *GNU flex-u* je to zelo učinkovito orodje za pridobivanje posebnih informacij iz podatkovnih tokov, kot so npr. elektronski naslov, naslov IP, številke kreditnih kartic ... Rezultati preizkušanja so vidni v tabeli 5. Ker orodje privzeto omogoča večnitno delovanje, se je pri testiranju skušalo definirati še optimalno število procesorjev, ki naj bi jih vsebnik z ekstraktorjem imel na voljo. Presenetljivo pa se je vseeno najbolje izkazala varianta z eno nitjo in enim procesorjem. Tako se npr. bolje obnese 48 enojedernih instanc, kot pa ena instanca z 48 jedri. Sicer pa se je za ta test uporabilo 500 MB podatkov iz že omenjenega GovDoc korpusa [8].

Št. vsebnikov	4	12	24	48	96	192
MB/s	3.5	17.9	22.7	54.8	59.4	151.5
MB/s na vseb.	0.9	1.5	0.9	1.1	0.9	1.2

Tabela 5: Pretok masovnega ekstrahiranja v odvisnosti od števila vsebnikov



Slika 3: Skalabilnost forenzičnih orodij z uporabo vsebnikov

5.6 Indeksiranje metapodatkov datotečnega sistema

Kot že omenjeno je v SCARF-u za namene shranjevanja metapodatkov forenzične tarče in rezultatov uporabljen *ElasticSearch (ES)*. Za preizkus delovanja le-tega je bilo pridobljenih in razčlenjenih 200 GB metapodatkov iz NTFS slike. Ker je ES namenjen distribuiranem izvajanju, se je testiralo dve različni arhitekturi, in sicer eno samo ES vozlišče in distribucijo sedmih vozlišč namenjenih opravljanju različnih nalog. V tabeli 6 je vidno, da uporaba namestitve v gruči pomeni super-linearno izboljšanje hitrosti v primerjavi s samostojno namestitvijo.

Št. vsebnikov	1	7
Št. zapisov/s	1299	14236

Tabela 6: ElasticSearch pretok v primeru samostojne konfiguracije in konfiguracije v gruči

5.7 Povzetek

Pretok obdelanih podatkov v odvisnosti od števila vsebnikov vseh testiranih orodij je prikazan še na sliki 3. Le ta prikazuje zelo dobro skaliranje orodij *ExifTool* in *Bulk Extractor*. Pri globoki nevronski mreži *OpenNSFW* pride do precej manjšega povečanja pretoka, a je povečanje vseeno vidno. Pretok pri *Apache Tiki* pa se veča zelo počasi, kar nakazuje ne-skalabilno arhitekturo orodja. Združeni rezultati testiranj so zbrani v tabeli 7.

6. ZAKLJUČEK

V članku [13] so tako raziskali uporabo vsebnikov za namene povečanja učinkovitosti orodij za digitalno forenziko. Rezultati testiranja kažejo na to, da tak način omogoča soočanje z vedno večjo količino forenzičnih podatkov, ki so potrebni obdelave. Pokazano je namreč bilo, da že gruča s štirimi vozlišči omogoča visoko povečanje pretoka obdelave podatkov. Ker je večino opravil, ki se jih je preizkusilo, mogoče paralelizirati, so v veliki meri testi pokazali linearno ali skoraj linearno skaliranje, kar kaže na to, da bi z večjo gručo, sestavljeno iz približno 20-40 vozlišč, lahko opravili večino trenutnih forenzičnih opravil. Zanimivo bi bilo tudi ugotoviti, kako bi se izkazala namestitev vsebnikov v javne oblake, kot so npr. AWS, Azure, Google Cloud, vendar se tu pojavljajo vprašanja o tem, ali je pametno forenzične podatke nalagati v javne oblake. Prav tako je bilo v delu pokazano, da uporaba vsebniških tehnologij omogoča enostavno dodajanje novih funkcionalnosti k obstoječemu okolju.

Zaključi se lahko, da uporaba vsebnikov zagotavlja učinkovito platformo, ki rešuje problema skalabilnosti in enostavne razširljivosti z novimi funkcionalnostmi.

7. Literatura

[1] Apache tika. Dostopno na `http://tika.apache.org/`. [Dostopano 12.05.2018].

[2] Exiftool by phil harvey. Dostopno na `https://sno.phy.queensu.ca/~phil/exiftool/`. [Dostopano 12.05.2018].

[3] Open nsfw model. Dostopno na `https://github.com/yahoo/open_nsfw`. [Dostopano 12.05.2018].

[4] Institute, n. f. hansken. Dostopno na `https://www.forensicinstitute.nl/products_and_services/forensic_products/hansken.aspx`, 2016. [Dostopano 07.05.2018].

[5] S. Alba. Deploy and manage any cluster manager with docker swarm. Dostopno na `https://blog.docker.com/2015/11/deploy-manage-cluster-docker-swarm/`, 2015. [Dostopano 12.05.2018].

[6] B. Carrier. The sleuthkit (tsk) and autopsy: Open source digital forensics tools. Dostopno na `http://sleuthkit.org`. [Dostopano 12.05.2018].

[7] B. des Ligneris. Virtualization of linux based computers: the linux-vserver project. In *19th International Symposium on High Performance Computing Systems and Applications (HPCS'05)*, pages 340–346, May 2005.

[8] S. L. Garfinkel. Digital media triage with bulk data analysis and bulk$_e$*xtractor*.*ComputersSecurity*, 32 : 56 − −72, 2013.

[9] D. Inc. Swarm mode overview. Dostopno na `https://docs.docker.com/engine/swarm/`. [Dostopano 12.05.2018].

[10] W. R. Kamp, P.-H. Jails: conning the omnipotent root. in: Pro-ceceedings of the second international system administration and networking conference. Dostopno na `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.118.3596&rep=rep1&type=pdf`, 2000. [Dostopano 12.05.2018].

[11] C. Miller, D. Glendowne, D. A. Dampier, and K. Blaylock.

Forensicloud: An architecture for digital forensic analysis in the cloud. 2014.

[12] V. Roussev. Building open and scalable digital forensic tools. In *2011 Sixth IEEE International Workshop on Systematic Approaches to Digital Forensic Engineering*, pages 1–6, May 2011.

[13] C. Stelly and V. Roussev. Scarf: A container-based approach to cloud-scale digital forensic processing. *Digital Investigation*, 22:S39 – S47, 2017.

[14] O. o. t. i. g. U.S. Department of Justice. Audit of the federal bureau of investigation's philadelphia regional computer forensic laboratory. Dostopno na `https://oig.justice.gov/reports/2015/a1514.pdf.`, 2015. [Dostopano 12.05.2018].

DODATEK

V spodnji tabeli 7 so za lažjo primerjavo predstavljeni vsi rezultati testiranj ogrodja SCARF.

Operacija \Št. vsebnikov	**Količina**	**4**	**8**	**12**	**24**	**48**	**96**	**192**
Šifrirno zgoščevanje	MB/s	345	-	857	985	985	948	992
Ekstrahiranje metapodatkov	MB/s	5.2	17	-	99	151	170	192
	MB/s na vseb.	1.3	2.1	-	3.1	2.4	1.8	1.0
Klasifikacija slik	MB/s	0.4	1.4	2.5	3.8	7.2	10.9	21.3
	Datotek/s	0.8	2.2	3.9	7.1	13.4	20.3	38.5
Apache Tika	MB/s	0.5	-	1.1	2.4	3.5	5.8	6.7
	MB/s na vseb.	0.13	-	0.09	0.10	0.07	0.06	0.03
Masovna ekstrakcija	MB/s	3.5	-	17.9	22.7	54.8	59.4	151.5
	MB/s na vseb.	0.9	-	1.5	0.9	1.1	0.9	1.2

Tabela 7: Tabela primerjave različnih operacij z uporabo platforme SCARF, v odvisnosti od števila vsebnikov

Review of the paper: Availability of datasets for digital forensics – And what is missing

Stefan Ivanisević
63170405
University of Ljubljana
Faculty of Computer and Information Science
Ljubljana, Slovenia
si0539@student.uni-lj.si

Božen Jovanovski
63160400
University of Ljubljana
Faculty of Computer and Information Science
Ljubljana, Slovenia
bj9914@student.uni-lj.si

ABSTRACT

This paper is a review of the paper *Availability of datasets for digital forensics and what is missing*[1], with an overview of the wider area, the referenced papers and concluded with our own opinion of the matter. There are many challenges for researchers that require datasets in the field of Digital Forensics and we try to identify them and offer an overview of the whole situation based on the original paper. We discuss the availability of repositories, the problem with sharing data and we present the findings and results.

Keywords

Availability, Data collection, Dataset, Origin, Experiment generated, User generated, Repository

1. INTRODUCTION

While every day researchers are conducting new researchers, for some of them they need and for some they do not need datasets for performing an examination of data or making conclusions that are valid for their scenario. For successful contribution into a specific scientific area, researchers that are using datasets in their work should follow C. Graheda's at al. [1] citation "In order to produce high-quality research results, we argue that three critical features must be examined:

1. Quality of the datasets. This helps guarantee that results are accurate and generalizable. Researchers need data that is correctly labeled and similar to the real world or originates from the real world.

2. Quantity of the datasets. This ensures that there is sufficient data to train and validate approaches/tools which is especially important when utilizing machine learning techniques.

3. Availability of data. This is critical as it allows the research to commence and ensures reproducible results helping in improving the state of the art."

C. Greheda at al. analyzed 715 articles that are related to cyber security and forensics research form the years 2010-2015 and categorized the data's origin, analyzed its availability and examined the different kinds of datasets.

2. LIMITATIONS

All of the data that is analyzed is done manually. The analysis is done on datasets from papers in the range of a 6 years period from 2010 to 2015.

3. RELATED WORK

Inspiration for this research comes from Abt and Baier [2] who analyzed the availability of ground-truth in network security research. In this article, the main weaknesses that are facing cyber-security/forensics researchers are presented. Low reproducibility, comparability, and peer validated research. Other researchers that are cited in this paper have also stated that it is hard to validate one's study due to the non-availability of datasets that are used in their research or luck of standardized datasets.

4. METHODOLOGY

Focus in their study is on all kinds of datasets that can be useful in cyber-security/forensics research (malwares, disk images, memory dumps ...).

4.1 Definition of a dataset

They defined a valid dataset as a collection of related items for a specific scenario and were collected for experiment or analysis, but on the other hand they did not consider an input that was only used to measure runtime efficiency, results written to log files or a tool that outputs data which is never used as valid datasets.

4.2 Analyzing peer-reviewed articles

In this first phase of analyzing the availability of datasets they analyzed publications from digital forensics and security conferences and journal publications and for each article they asked four main questions:

1. **Origin of datasets:** What is the origin of datasets? They divided datasets into four groups how they can be created:

- *Computer generated (algorithm, bot, etc.)*
- *Experiment generated (user creates specific scenario)*
- *User generated (real world data)*
- *Mixed datasets (user, experiment & computer generated datasets)*

2. **Availability of datasets:** Are datasets available to the community?

3. **Kind of datasets:** What datasets exist and can be used by researchers?

4. **What is missing:** What datasets are currently missing?

The answers of the questions above, which the authors of the original paper found are presented in the paragraphs that follow.

4.3 Online searches

In the second phase, they searched Google for any datasets that were not found during the first phase while analyzing publications. Four queries related to forensics, cyber-security, and availability of datasets were searched and in each of them, the first 100 results were examined. The exact queries that are used are 'available digital forensics dataset repositories', 'available cyber-security and forensics dataset repositories', 'available malware dataset repositories', and 'available computer dataset repositories'. If the datasets/repositories that are found through Google search are used in the first phase then they are identified as a source of the datasets.

5. RESULTS OVERVIEW AND ORIGIN

Out of 715 articles that were analyzed approximately 49% used datasets in their research. The following conferences were analyzed:

- IEEE Security & Privacy (S & P) with 76 out of 240 ($\approx$ 49%) articles used datasets
- Digital Forensic Research Workshop (US & EU) with 78 out of 91 ($\approx$ 86%) articles used datasets
- International Conference on Digital Forensics & Cyber Crime (ICDF2C) with 60 out of 107 ($\approx$ 56%) articles that used datasets
- Association of Digital Forensics, Security & Law (ADFSL, Conference) with 29 out of 87 ($\approx$ 33%) articles that used datasets
- Digital Investigation (Journal) with 108 out of 190 ($\approx$ 57%) articles that used datasets

Overview of the origin is shown in **Table 1**.

Table 1: Overview of the origin of the 351 identified datasets out of the 715 analyzed articles.

Articles	Total	
Experiment generated	56.4%	198
User generated	37.7%	129
Computer generated	4.6%	16
Mixed sets (user, experiment & computer)	2.3%	8

5.1 Experiment generated datasets

This type of generating datasets is mostly used in research and there are two main reasons why researchers are using experiment generated datasets. The first one, which is the reason in most cases, is lack of real world datasets available that are fulfilling all requirements for their scenario. The second one is easier manipulation, testing and verifying data.

5.2 User generated datasets

The second most used type of datasets was the real word datasets but the main issue in using real world data was in copyrights and privacy laws that are forbidding sharing data. There can be multiple sources from which user generated datasets can be pulled, but the following ones are the most frequent:

- *Dataset was released:* The most used dataset of this type was the e-mail dataset posted online by the Federal Energy Regulatory Commission after its investigation but was eventually removed to avoid violating user privacy rights.
- *User data was collected before research:* Frequently used by larger institutions that collect data from interaction with their employees/students and perform the desired research.
- *Collaboration with law enforcement:* In their research, they stumbled upon eight collaborations between law enforcement agencies and academies.
- *Source of data is online:* Data that can be found online (e.g. YouTube, Google images, etc.)

5.3 Computer generated datasets

One of the examples for Computer generated datasets is in paper [3] where they utilized pseudo-random data from *SecureRandom.random_bytes* to analyze the precision & recall rates of approximate matching algorithms.

5.4 Usage of third party databases, services or online tools

About 20.4% (39/191) used third party databases, services or online tools to retrieve information (e.g. In the paper [4] Al-Shaheri et al. queried openMalware.org [1] to acquire malware for their research).

[1]http://openMalware.org

5.5 Availability of datasets

Availability of datasets is the second phase of analyzing datasets and availability and re-use of datasets is presented in **Table 2**. There are three main aspects of availability and re-use of datasets:

Table 2: Results of 715 analyzed articles with 351 containing datasets.

Articles	Total	
Created through research	45.6%	160/351
– Existed prior to research (re-use)	54.4%	191/351
Currently available sets	29.0%	102/351
– Existed and available (re-use)	50.3%	96/191
– Created and released	3.8%	6/160
Exist and not available	29.3%	56/191
Available as services	20.4%	39/191

5.5.1 Creating vs. re-using datasets

Evaluating performances / comparing algorithms that are created during research is often best to done on real world datasets and because of that the first row in **Table 2** seems reasonable. Out of 715 articles that they have analyzed 46.5% produced their own datasets while the other 54.4% used existing datasets. Even if usage of self-made datasets is high, the main reason for that is that datasets are usually not shared or existing datasets do not fit into the researcher's current scenario.

5.5.2 Currently available datasets

Only 29%(102) of all sets are shared with the researchers' community. Only 6 new datasets were created from 2010 to 2015 and the rest of the datasets, 96 already existed. Only 2% out of 102 datasets were computer generated datasets, 38.2% were experimentally generated and majority originated from four main repositories. More than half of the datasets 59.8% were real world datasets which originated from four different online repositories (Digital Corpora, Enron E-mail Dataset, the t5-corpus and Android Malware Genome Project (no longer available))

5.5.3 Non available datasets

C. Grajeda et al. [1] discovered that for 29.3% (56 191) articles' datasets were not shared/available. They organized this articles into three groups:

- *Source is unknown:* 22 out of 56 researchers failed to specify the origins of their datasets which brings questions about the quality and integrity of data the produced in their research.
- *Source has privacy restrictions:* With 26 out of 56 articles this puts this group as the major problem for unavailability of datasets.
- *Source not accessible:* 1 of 7 articles had problems with accessibility of their datasets such as temporarily unavailable, download link broken or not maintained anymore. This can be a huge problem in the future.

5.6 Kinds of datasets

In this section, we describe the different datasets the authors of the paper found and we will give an overview of what was found in their research. They found over 70 different datasets through their article analysis and they organized them into 21 categories. We will give an overview of the major ones in the subsections below.

5.6.1 Malware datasets (computer and mobile)

The authors say that they found seven real world data online repositories of computer and mobile malware samples, three for android mobile malware and four for computer malware.

First we will go through the Android malware repositories:

1. Drebin is a collection of 5560 Android samples from 179 different malware families collected between 2010 and 2012 and used in the paper by Talha [5].
2. Contagio Mobile Mini-Dump is part of a larger malware repository called Contagio Malware Dump which functions more like a blog than a classic repository. Users can download the repository, but they can also extend it by uploading. There are over 200 malware posts posted from 2011 to 2016 and they might contain more than one malware sample per post.
3. In the paper *Andro-AutoPsy: Anti-malware system based on similarity matching of malware and malware creator-centric information*[6] a dataset of 9990 malware samples was used and this dataset can be requested for research purposes.

Regarding computer malware the researchers found four repositories used in the articles they analyzed:

1. Contagio Malware Dump has around 400 posts.
2. VX Haven is a virus information website that contains over 271 000 computer malware samples. The authors mention that there is no information on how often this website is updated.
3. Virus Share has a collection of over 27 million malware samples of both mobile and computer malware. This site is monthly updated and the dataset can be obtained only through an invitation which means that the admin of the site decides to grant or deny access to all who request it.
4. KernelMode.info is a forum with posts from 2010 to 2016 and according to the authors appears to still be active, however, the number of malware samples cannot be verified.

5.6.2 E-mail datasets

Three e-mail datasets were found by the authors. The Enron E-mail Dataset version 2015 which contains over 619 000 real world e-mail between 158 users, the Apache online e-mail repository with around 76 000 real world e-mails

which was not meant to be a dataset and in the end the authors say they found 12 e-mails in Digital Corpora's experiment generated scenarios, but those e-mails were never used.

5.6.3 File sets/collections

According to the authors, the most prominent and comprehensive dataset is GovDocs1 corpus from Digital Corpora which consists of approximately 1 million documents obtained by crawling the .gov domain. Two subsets of this massive dataset were created by researchers for their papers, one is t5, which contains 4457 files of various types and msx-13 which contains 22 000 MS Office 2007 user generated random files.

5.6.4 Ram Dumps

Six repositories with over 90 dumps were found by the authors. The biggest and the most comprehensive one is from Digital Corpora with 88 samples and over 44 GB size. A set was published by Minnard[7] where the authors acquired their own RAM data and they made a sample of 1 GB available for download, the rest can be obtained by request. Another set of five 1 GB RAM dumps is provided by the CFReDS Project. Two dumps of WinXP 32-bit machines were made available by the DFRWS' forensic challenge. There is a dataset in The Art of Memory Forensics book and it can be downloaded from the appropriate site.

5.6.5 Images of computer drives

Two sets were found by the authors, the first is from Digital Corpora and it is called the Real Data Corpus which in 2011 contained 1289 hard drive images. The second and much smaller set is provided by the CFReDS Project and it contains three images extracted with different tools.

5.6.6 Images of other devices

Beside hard-drive, the authors found other images such as:

1. Cell phones - 26 images from two repositories, CFReDS and Digital Corpora
2. Gaming systems - only 2 sets of Xbox images were identified, the first was released from Moore[8] and the second came from the nps-2014 XBox-1 scenario.
3. SIM card - 3 images were discovered in CFReDS but they were never used.
4. Apple iPod & Tablet - Digital Corpora offers 10 iPod disk images and 24 various tablet images but these were not used anywhere.
5. Flash Drives - 643 flash images were found in Digital Corpora and two sets called nps-2009-canon2 and nps-2013-canon1 of 32 MB SD cards.

5.6.7 Network traffic

A few sets were discovered by the authors regarding network traffic information. The first set was generated for the DFRWS 2009 forensic challenge and contains PCAP files where most of the traffic is through HTTP on port 80. The second set was created by Karpisek[9] and it contains 3 PCAP files containing WhatsApp register and calls traffic. The final repository is a site called CRAWDAD which contains different types of wireless network traffic.

5.6.8 Others

The authors also found sets of scenarios/cases for analysis and other datasets such as: pictures, language corps, chat logs and passwords. The previous datasets were discovered or extracted from scientific articles. The authors also used a Google search and found ten sources providing datasets either through links to other websites or directly available. They can be found on the website of the authors of the original paper http://datasets.fbreitinger.de/.

6. DISCUSION AND CONCLUSIONS

6.1 What is missing?

According to the authors, the results from their research show that researchers do not like to share their datasets. They suggest several reasons why this would be the case. One possibility is that researchers do not have the capability for sharing the data at the time of publishing. Another could be privacy concerns that researchers might have so that they are not sure if sharing the datasets further is allowed or even legal. Thirdly, the authors say that researchers might not be aware of the importance of their data, and finally, there could be some intellectual property reasons. There is also a lack of datasets regarding investigations concerning cloud computing.

The original study has shown that there is missing a variety of datasets. So for an example, the authors did not find samples for Play Station 4 gaming console, or smart TVs. And from the datasets that are available, there is a huge difference in numbers, for an example, the malware samples count to around 27 million, however, there are only 26 available smartphone images.

Many of the available datasets are not updated, which in some cases can make a huge difference, for an example with malwares. The authors propose some sort of a single centralized, maintained and organized repository which would be managed by the community.

Considering the privacy issues, the authors propose that research should be done with de-identification of data, which means that data should be un-personalized before using it. They also suggest that it could be useful if researchers are forced to share their data as well as their findings.

6.2 Conclusion

After reviewing the original paper and some of the references and other work on this topic it is obvious that the main problem in the Computer Forensics department is the absence of data sharing. The results that the authors provided show us that less than 4% of the researchers in the community shared their data. Combined with the outdated datasets that are available, the privacy issues and the lack of variety of data it is fair to say that there are many difficulties for researchers who require data repositories.

The authors propose some solutions such as a central-

ized community based repository which would definitely help with the data availability. We believe that there should be a change in the culture of the community, where researchers are stimulated to share their data in a safe manner. Perhaps, in time, regulations and rules that obligate researchers to act responsibly in consideration with sharing the data can be implemented by appropriate institutions to achieve these goals. In conclusion, we believe that there is more research to be done regarding this topic and this study is definitely a step in the right direction.

7. REFERENCES

[1] Cinthya Grajeda Mendez, Frank Breitinger and Ibrahim Baggili: Availability of Datasets for Digital Forensics - and What is Missing. DFRWS USA 2017 https://www.sciencedirect.com/science/article/pii/S1742287617301913

[2] Abt, S., Baier, H., 2014. Are we missing labels? A study of the availability of ground-truth in network security research. In: Proceedings of the 3rd Workshop on Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS'14) (September 2014). IEEE. https://www.sciencedirect.com/science/refhub/S1742-2876(17)30191-3/sref1

[3] Breitinger, F., Stivaktakis, G., Roussev, V., 2014c. Evaluating detection error tradeoffs for bytewise approximate matching algorithms. Digit. Investig. 11, 81e89. https://www.sciencedirect.com/science/article/pii/S1742287614000450?via%3Dihub

[4] Al-Shaheri, S., Lindskog, D., Zavarsky, P., Ruhl, R., 2013. A forensic study of the effectiveness of selected anti-virus products against ssdt hooking rootkits. In: Proceedings of the Conference on Digital Forensics, Security and Law, pp. 137e160. https://www.scopus.com/record/display.uri?eid=2-s2.0-85026763745&origin=inward&txGid=041bcd1542550bb1a567e20073850196

[5] Talha, K.A., Alper, D.I., Aydin, C., 2015. Apk auditor: permission-based android malware detection system.

[6] Jang, J.-w., Kang, H., Woo, J., Mohaisen, A., Kim, H.K., 2015. Andro-autopsy: anti-malware system based on similarity matching of malware and malware creator-centric information. Digit. Investig. 14, 17-35.

[7] Minnaard, W., 2014. Out of sight, but not out of mind: traces of nearby devices' wireless transmissions in volatile memory. Digit. Investig. 11, S104-S111.

[8] Moore, J., Baggili, I., Marrington, A., Rodrigues, A., 2014. Preliminary forensic analysis of the xbox one. Digit. Investig. 11, S57-S65.

[9] Karpisek, F., Baggili, I., Breitinger, F., 2015. Whatsapp network forensics: decrypting and understanding the whatsapp call signaling messages. Digit. Investig. 15, 110-118.

Selektivno brisanje podatkov

Predstavitev problema selektivnega brisanja podatkov in pregled širšega področja ter obstoječih rešitev

Rok Lampret
Fakulteta za računalništvo in informatiko
Večna pot 113
Ljubljana, Slovenia
rl7811@student.uni-lj.si

Jan Šubelj
Fakulteta za računalništvo in informatiko
Večna pot 113
Ljubljana, Slovenia
js3445@student.uni-lj.si

POVZETEK

Vse več forenzičnih preiskav dandanes vključuje tudi zaseg digitalnih naprav. Večje ali manjše količine podatkov na zaseženih napravah so zasebne narave. Z vse strožjimi zakoni o varstvu osebnih podatkov in zasebnega življenja se digitalni forenziki v Evropi in po svetu srečujejo s problemom kako takšne podatke zares uničiti, obenem pa s tem ne kvariti verodostojnosti dokazov. Postavimo se tudi na drugi breg, kako bi lahko orodja za selektivno brisanje podatkov uporabili za anti-forenzično delovanje. Recimo da je bil vohun razkrinkan in se mora hitro znebiti občutljivih podatkov. Pojavi se problem kako se podatkov znebiti na nesumljiv način. Nazadnje še predstavimo primer implementacije sistema za selektivno brisanje podatkov na NTFS datotečnih sistemih. Ta je bil implementiran za skupek orodij Digital Forensic Framework.

Ključne besede

datotečni sistemi, digitalni dokazi, digitalna forenzika, osebni podatki, selektivno brisanje

1. UVOD

Na uporabnost selektivnega brisanja lahko gledamo iz dveh glavnih zornih kotov. Prvi je forenzični, drugi pa anti-forenzični vidik. V nadaljevanju so razlogi za uporabo selektivnega brisanja iz obeh vidikov bolj podrobno predstavljeni, vendar bi na kratko lahko povzeli, da:

- iz forenzične perspektive, potrebujemo selektivno brisanje predvsem zato, da ugodimo strogim zakonom o varstvu osebnih podatkov in pravici do zasebnega življenja
- iz anti-forenzične perspektive zato, da lahko varno (in hitro) izbrišemo obremenilne dokaze.

Največkrat ni problem izbrisati vsebine datotek, za to obstaja že vrsto bolj ali manj zanesljivih orodij. Problem ponavadi nastane pri brisanju metapodatkov, ko želimo popolnoma zabrisati sledi o tem, da je ta izbrisana datotetka nekje, nekdaj sploh obstajala.

1.1 Vsebina

V poglavju 2 predstavimo širši problem selektivnega brisanja tako iz forenzičnega kot anti-forenzičnega vidika. V poglavju 3 predstavimo alternativo selektivnemu brisanju pri forenzičnem delu, to je selektivno kopiranje. V poglavju 4 predstavimo Zoubekovo[8] implementacijo orodja za selektivno brisanje na NTFS datotečnih sistemih. V poglavju 5 na kratko predstavimo smer nadaljnjega razvoja področja.

2. PREGLED PODROČJA

2.1 Forenzični vidik

Veliko kriminalističnih preiskav, tudi takih, ki niso same po sebi digitalne narave, vključuje zaseg digitalnih naprav. Danes, v času vsesplošne digitalizacije, lahko raznolike digitalne dokaze najdemo na vrsti digitalnih medijev kot so računalniki, telefoni in različne vrste pomnilniških medijev. Ustvarjanje tako imenovanih *slik* (angl. *image*) - popolnih kopij nosilcev digitalnih podatkov je postalo že stalna praksa v forenziki za zagotavljanje forenzične trdnosti dokazov.

Potreba po orodjih za selektivno brisanje (nerelevantnih) podatkov izhaja predvsem iz zakonov, ki omejujejo dostop do informacij zasebne narave in njihovo uporabo. Ena izmed takih pravic je zapisana v 8. členu Evropske konvencije o človekovih pravicah (EKČP)[3]– Pravica do spoštovanja zasebnega in družinskega življenja:

> Vsakdo ima pravico do spoštovanja svojega zasebnega in družinskega življenja, svojega doma in dopisovanja. Javna oblast se ne sme vmešavati v izvrševanje te pravice, razen če je to določeno z zakonom in nujno v demokratični družbi zaradi državne varnosti [...]

Še posebej stroge zahteve glede zasega podatkov in njihove uporabe za kriminalistično preiskavo imajo nemški organi kazenskega pregona. Člen 100a podpoglavje 4 nemškega zakona o kazenskem postopku[5] pravi:

> [...] Informacije, ki se tičejo zasebnega življenja [...] ne bodo uporabljene. Kakršnikoli zapisi o le-teh bodo nemudoma izbrisani. Dejstvo, da so bile pridobljene in izbrisane, bo zabeleženo.

Po pričevanju slovenskega policista - digitalnega forenzika, se slovenska policija s takšnimi problemi še ne srečuje. V slovenskem Zakonu o kazenskem postopku[7] člena 219.a in 223.a govorita o zasegu elektronskih naprav, vendar takšnih zahtev, glede brisanje zasebnih podatkov, kot nemški člen 100a ne postavljata.

V praksi je zaseg izključno relevantnih podatkov težko izvedljiv, zato bi lahko orodja za selektivno brisanje pripomogla k izpolnjevanju prej omenjenih in drugih zakonov o varstvu zasebnih podatkov. Trenutno sicer še ni splošno uveljavljenega orodja za brisanje nerelevantnih podatkov, tako med samim zasegom kot potem.

Glavni razlog proti uporabi takih orodij je, da brisanje podatkov vedno spremeni dokaz. Tukaj naletimo na prvi problem, spreminjanje dokaza lahko uniči njegovo verodostojnost in s tem uporabnost na sodišču. Po drugi strani pa lahko kršimo katerega od zakonov če dokazov ne obdelamo pravilno.

2.2 Anti-forenzični vidik

Castiglione et al.[2] navajajo vrsto razlogov za uporabnost orodja za selektivno brisanje podatkov iz anti-forenzičnega vidika. Obstajajo scenariji, kjer bi si želeli, da lahko hitro in v popolnosti izbrišemo sledi določenih podatkov na nekem sistemu. Npr. da so odkrili vohuna, ki mora na nesumljiv način odstraniti občutljive podatke, ali pa političnega disidenta, ki mu na vrata trka policija. V takšnih in podobnih situacijah se lahko znajdejo tudi še novinarji, žvižgači ipd.

Očitno je da, ima anti-forenzični vidik nekaj dodatnih omejitev oz. pričakovanj od orodja za selektivno brisanje. Predpostavljamo tudi še, da se takšno orodje z anti-forenzičnim namenom uporablja na živem sistemu. Dodatna je še časovna omejitev. Vrsto scenarijev zahteva (npr. policijska racija), da so občutljive informacije izbrisane z enim miškinim klikom v nekaj sekundah. Želimo tudi, da je orodje sposobno *samouničenja*. Orodje za brisanje naj po končanem delu samo sebe izbriše iz sistema in zakrije vse sledi o svoji aktivnosti. To pomeni tudi pobrisati sledi v glavnem pomnilniku. Castiglione et al.[2] za takšno nalogo predlagajo interpretirane jezike, saj moderni operacijski sistemi programom močno otežijo spreminjanje lastne kode. Postopek uničenja so avtorji tudi implementirali in sicer v Javi. Prevedeni Java programi se posredno izvajajo v JVM (*Java Virtual Machine*), kar jim tako omogoča spreminjajo lastno datoteko, iz katere se izvajajo.

Po končanem delu želimo, da forenzična analiza diska ne vzbuja nobenega suma. Ravno zato delamo selektivno brisanje in ne brisanje celotne particije ali diska (angl. *wiping*). Veliki odseki samih ničel na disku lahko nakazuejejo na to, da je nekdo tam brisal podatke. Veliko bolj zaželeno je, da bi lahko posamezne datoteke brisali tako, da za seboj ne pustijo niti sledi o tem, da se je brisanje zgodilo. To vključuje brisanje ali manipulacijo metapodatkov in bolj pametno brisanje vsebine, kot samo prepisovanje z ničlami. Lahko bi vsebino prepisali z naključnimi podatki, vendar je visoka stopnja entropije lahko tudi sumljiva. Morda je bolje prepisati vsebino s podatki iz katerega drugega dela diska.

Vidimo, da potreba po orodjih za selektivno brisanje obstaja, tako iz forenzičnega kot anti-forenzičnega vidika. Obenem pa, da brisanje podatkov ni tako enostavno še posebej pa ne na modernejših operacijskih in datotečnih sistemih.

2.3 Brisanje podatkov

Pogosto operacijski sistemi, kot so Windows in macOS, brišejo datoteke tako, da zgolj izbrišejo ali označijo z zastavico vnos o tej datoteki v tabeli datotek npr. v *Master File Table* ($MFT) na datotečnih sistemih NTFS. Sama vsebina datoteke (in metapodatki) pa lahko ostanejo bolj ali manj nedotaknjeni na disku, dokler jih ne prepišejo neki novi podatki. Tak način brisanja datotek ni primeren za (anti-)forenzično uporabo, saj obstaja vrsto orodij, ki znajo tako izbrisane datoteke restavrirati.

Podobno tudi formatiranje diska običajno zgolj prepiše particijske tabele, medtem ko podatki še vedno ostanejo na disku. Orodja za klesanje datotek, znajo poiskati in restavrirati vsebino tako pobrisanih datotek, ne pa tudi metapodatkov. Slednje je sicer dobro iz anti-forenzičnega vidika, vendar ima formatiranje in prepisovanje celotnih particij dva druga potencialna problema. Prvi je ta, da izbrisane celotne particije ali diski lahko vzbudijo sum pri forenzični analizi in drugi je to, da lahko ta postopek vzame preveč časa.

Da onemogočimo možnost klesanja izbrisanih datotek, lahko disk formatiramo tako, da vse podatke prepišemo z naključnimi podatki ali ničlami. Nekatera orodja to naredijo tudi pri brisanju posameznih datotek. Vendar navadno taka orodja prepišejo samo vsebino te datoteke, ne pa tudi metapodatkov. V primeru, da metapodatki ostanejo na disku, lahko iz njih še vedno izvemo informacije o uporabniku. Na primer, predstavljajmo si, da je bila oseba, kateri so zasegli disk, malo pred tem na počitnicah in da je shranil vse slike iz dopusta na zaseženi disk. Iz same strukture imenikov in imen, npr. “Poletje 2017 - Jerevan”, lahko izvemo marsikaj o osumljenčevem zasebnem življenju.

Iz anti-forenzičnega vidika pa je še posebej pomembno, kako se lotimo tudi prepisovanja vsebine datotek. Najbolj naiven pristop s pisanjem ničel je sicer efektiven in najhitrejši, omejen je namreč le s pisalno hitrostjo diska, vendar pa lahko na disku vzorci, kot je npr. dolgo zaporedje ničel, pri forenzični analizi vzbudijo sum, da je nekdo brisal podatke. Ravno tako je lahko sumljivo, če podatke prepišemo z naključnimi števili. Da dosežemo dovolj veliko stopnjo entropije bi potrebovali kriptografsko močan (pseudo) naključen generator števil, kar pa zna narediti tak postopek brisanja dokaj počasen. V nasprotnem primeru, če “naključna” števila niso dovlj naključna, se lahko z globoko analizo ugotovi, da so bila števila generirana z algoritmom[2]. V obeh primerih, če je naključnost preslaba oz. je stopnja entropije atipično velika, pa lahko forenzik posumi, da se je tu zgodilo brisanje podatkov, kar naredi ta postopek manj idealen, kot se morda zdi na prvi pogled.

Verjetno manj sumljiva metoda bi lahko bila prepisovanje

podatkov z že obstoječimi podatki nekje drugje na disku. Dandanes, ko razpolagamo z večjimi količinami prostora za podatke, namreč ni nič čudnega, da so datoteke na disku podvojene ali potrojene (varnostne kopije, predpomnjenje ipd.). Največji pomanjkljivosti tega načina sta, da je počasen in kako določiti katere podatke naj uporabimo za prepis. Iz anti-forenzičnega vidika morda tudi ni vedno nujno potrebno brisanje kot tako. Če bi znali vsebino datoteke samo preoblikovati na tak način, da ne razkrije svojega pravega pomena in obenem nova vsebina ne vzbuja suma, potem je namen ravno tako dosežen.

V datotečnih sistemih kot je NTFS so metapodatki o datotekah shranjeni v posebnih podatkovnih strukturah, nekaj jih je shranjenih neposredno v $MFT. Selektivno brisanje v NTFS datotečnem sistemu se izkaže za težavno, saj lahko brisanje vseh podatkov, vključno z metapodatki, pokvari datotečni sistem. V takih primerih je primerneje, da se nekatere metapodatke zgolj ustrezno priredi in ne dejansko izbriše. Npr. moderni datotečni sistemi za pohitritev dostopnega časa do datotek, shranjujejo imena datotek v B drevesih. Zgolj grobo brisanje imena datoteke iz takšne podatkovne strukture jo lahko hudo pokvari in pusti dele drevesa nedostopne.

Zelo pomembna zahteva pri selektivnem brisanju je tudi verodostojnost preostanka podatkov. Poleg samega brisanja nerelevantnih podatkov (in pripadajočih metapodatkov) je potrebno zagotoviti, da preostali podatki ostanejo nespremenjeni. Če želimo, da je dokaz sprejemljiv na sodišču moramo znati pokazati, da relevantni podatki niso bili spremenjeni na noben način. K temu lahko pripomoreta dva pristopa. Prvi je natančno in dosledno dokumentiranje vseh sprememb in brisanj podatkov, drugi pa je dokazovanje verodostojnosti preostalih podatkov z Merklovimi drevesi[1].

Dodaten problem pri selektivnem brisanju predstavljajo tudi duplikati. Kot že rečeno, danes ni neobičajno, da si shranimo več kopij datotek. Tako lahko forenzik izbriše eno datoteko, nevedoč da se enaka datoteka nahaja še nekje drugje na disku. Ročno skrbeti za vse duplikate je lahko zamudno delo, ki ima veliko prostora za napake. Priročno bi bilo, če bi orodje za selektivno brisanje znalo tudi avtomatsko poiskati duplikate.

Selektivno brisanje je različno zahtevno tudi glede na tip podatkov, ki jih želimo izbrisati. Brisanje recimo programske opreme zna biti zelo težavno, saj lahko programi puščajo sledi o svojem obstoju na različnih mestih na sistemu in v različnih podatkovnih strukturah, kot je na primer Windows register in razne zabeležke.

2.4 Pričakovanja

Predstavili smo vrsto problemov pri brisanju podatkov. Upoštevajoč te probleme pričakujemo od orodja za selektivno brisanje, ki je primerno za forenzično delo, vsaj naslednje:

- da izbriše vso vsebino podatkov v popolnosti
- da najde in izbriše pripadajoče metapodatke
- da pusti datotečni sistem v delujočem stanju
- da natančno zabeleži vsako spremembo in brisanje
- da zagotovi verodostojnost preostalih podatkov

Predpostavljamo, da orodje za forenzično delo operira nad slikami diskov, medtem ko za anti-forenzično orodje predpostavljamo, da dela z živim sistemom.

Podobna pričakovanja imamo tudi od orodja za selektivno brisanje za anti-forenzično delovanje. Od tega orodja sicer ne pričakujemo oz. niti ne želimo, da natančno zabeleži vse spremembe in brisanja. Pričakujemo pa dodatno:

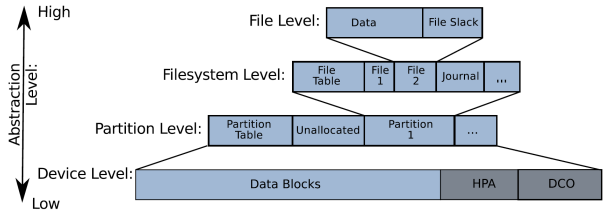
- da je dovolj hitro[1]
- da zna bolj napredno kot samo z ničlami prepisati staro vsebino
- da za sabo ne pušča nobenih sledi in je torej sposobno izbrisati tudi samo sebe (samouničenje).

3. ALTERNATIVA – SELEKTIVNO KOPIRANJE

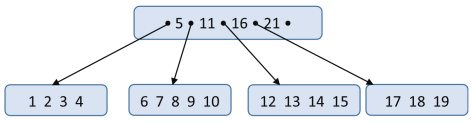
Stuttgen et al.[4] so razvili orodje za selektivno kopiranje (delanje delnih slik diskov), ki je lahko alternativa selektivnemu brisanju pri forenzičnem delu. Namesto, da naredimo sliko celotnega diska, že vnaprej naredimo izbiro, katere dele diska bomo skopirali. To pomeni, da moramo disk predhodno pregledati in se odločiti na mestu, kaj je za primer relevantno in kaj ne. To ima svoje prednosti in slabosti. Največja prednost je to, da lahko drastično zmanjšamo količino prekopiranih podatkov. Npr. kopiranje diska velikosti 2TB s hitrostjo 70MB na sekundo (hitrost modernega orodja za delanje slik diskov) vzame več kot 8 ur. To pomeni zmanjšanje časa, ki ga potrebujemo, da sploh začnemo z analizo. In še potem, ko analizo začnemo, je količina podatkov, ki jih mora forenzik pregledati, lahko znatno manjša. Še ena prednost je in sicer da se število bralnih dostopov do originalnega diska zmanjša. V primeru, da je disk v zelo slabem stanju in lahko odpove med celotnim kopiranjem, je smiselno kopirati le najbolj relevantne podatke. Največja slabost pa je seveda, da morda ni takoj očitno kaj vse je relevantno za primer in tako prekopiramo premalo podatkov. Zelo pomembno je tudi, da orodje poleg vseh izbranih podatkov prekopira še vse pripadajoče metapodatke.

Do sedaj smo se pri orodjih za selektivno brisanje pogovarjali večinoma na nivoju datotek in datotečnih sistemov, pri selektivnem kopiranju pa je pomembno, da znamo operirati še na drugih nivojih abstrakcije. Tukaj si želimo delovati tudi na nivoju particij in na najnižjem nivoju – nivoju diska. Tako na primer želimo skopirati nealocirani del diska in ga kasneje pregledati za morebitne skrite podatke. Stuttgen et al.[4] so v svoji implementaciji orodja dopustili veliko stopnjo granularnosti, kar pomeni, da lahko zelo selektivno izberemo kaj želimo kopiratni npr. samo datotečni *slack*.

[1]Sicer je res, da si v vsakem primeru želimo nasploh čim hitrejša orodja, vendar iz anti-forenzičnega vidika obstajajo scenariji, kjer je čas kritičnega pomena.



Slika 1: Različni nivoji abstrakcije[4].



Slika 2: Enostavno B-Drevo [8]

4. IMPLEMENTACIJA

Prototip sistema je bil kreiran kot vtičnik za forenzični skupek orodji Digital Forensic Framework (DFF). Sistem trenutno podpira le NTFS datotečni sistem, za katerega so se avtorji odločili zaradi popularnosti. Konceptualno sistem deluje tako, da uporabnik označi datoteke ali mape za brisanje, orodje izračuna kontrolno vsoto diska, pregleda disk, če obstajajo duplikati datotek, označenih za brisanje, ter nato izbriše podatke na disku nato zopet poračuna kontrolno vsoto in preveri če so bile spremenjene le datoteke označene za brisanje. Prav tako orodje izbriše meta podatkovni vnos izbrisanih datotek v glavni tabeli datotek ($MFT), saj lahko že samo ime datoteke nosi določene informacije. Po izbrisu orodje zopet izračuna kontrolno vsoto. Zaradi zagotavljanja forenzične trdnosti sistem celotni postopek tudi zabeleži. Orodje za izbirno brisanje je sestavljeno iz petih modulov, katere bomo opisali v nadaljevanju in povzemajo implementacijo opisano v [8].

4.1 NTFS in B-drevo

Za lažje razumevanje bomo v tem razdelku poskusili bralcu razložiti kako delujejo B-drevesa v NTFS sistemu ter odgovoriti, zakaj ni zadosti, da bi vnose samo izbrisali. V NTFS datotečnem sistemu so imena datotek shranjena v B-drevesih. Tako shranjevanje pospeši dostopni čas do datotek. Vsako vozlišče v drevesu je predstavljeno z gručo določene velikosti v kateri imena predstavljajo pozicijo vnosa v drevesu. Na sliki 2 je enostavno B-drevo. Vsak vnos ima tudi zastavico, ki določa ali je poddrevo priključeno na ta vnos. B-drevo je uravnoteženo in urejeno, zato so vsi vnosi v poddrevesu vedno leksikografsko manjši kot vnos starša.

Brisanje datotečnega imena v takem drevesu lahko uniči dostop do delov drevesa. Na primer, če bi izbrisali vrednost 5 v glavi drevesa, potem ne bi mogli več dostopati do vnosov od 1 do 4, saj noben starš več ne kaže na te vnose. Ob klicu na vnose od 1 do 4 bi se algoritem po drevesu spustil pri vnosu 11, ta pa kaže samo na vnose od 6 do 10, kar pomeni da klic na vnose od 1 do 4 ne bi vrnil nič. V podpoglavju 4.2.5 je prikazan bolj varen način brisanja imen v B-drevesu.

4.2 Implementirani moduli

V tem delu bomo opisali module ki sestavljajo vtičnik za izbirno brisanje. Ti moduli so:

- Izbirnik (Selector)
- Ujemnik (Matcher)
- Izklesovalec-Čistilec (Carver-Cleaner)
- Hashkalkulator (Hashcalculator)
- Izbrisovalnik (Deletion module)

4.2.1 Izbirnik

Izbirnik ali Selector, kot so ga poimenovali avtorji, je namenjen interpretaciji particijskih tabel. Z njim lahko uporabnik pridobi datoteke, ki so bile pobrisane tako, da se v $MFT pobriše zastavica v uporabi (in use).

4.2.2 Ujemnik

Ujemnik ali Matcher je modul, ki skrbi da je datotečni sistem na disku po brisanju še vedno koherenten. V $MFT imamo lahko več povezav, ki kažejo na isti datotečni blok. S tem se prepreči nepotrebno kopiranje identičnih podatkov ter s tem izboljšuje življensko dobo trdega diska. Če pri brisanju označimo tako datoteko, potem lahko privede do tega, da je v $MFT še vedno shranjena povezava do datoteke, ker pa je bil ta blok zbrisan, nam sistem javi napako datotečnega sistema. Modul ujemnik zato preišče $MFT, ali obstaja še kakšna povezava do datoteke, ki jo želimo zbrisati, ter jo nato označi temu primerno.

Dodatna funkcija ujemniškega modula je, da najde prave duplikate datotek. Ker uporabniki iz varnosti velikokrat hranijo iste datoteke na različnih particijah ali diskih, je treba te podatke vse locirati in izbrisati. Da bi uporabnik sam poskusil poiskati vse te duplicirane datoteke je težko, saj je že sama velikost trenutnih diskov problem in je velika možnost da bi uporabnik spustil kakšen duplikat. Zato so avtorji vtičnika implementirali v modul kalkulator varnostnih vrednosti, ki izračuna varnostno vsoto nad prvimi parimi bloki datoteke, ki je označena za brisanje, nato pa primerjajo to vrednost z varnostno vsoto prvih parih blokov vseh datotek, ki so enake velikosti kot označena.

Poleg opisanih funkcionalnosti modul tudi zabeleži zakaj je označil vse povezave (vnose v $MFT) in duplikate v beležko. S tem lahko uporabnik preveri in če je potrebno rekonstruira zakaj so bile povezave in duplikati označeni za brisanje.

4.2.3 Izklesovalec-Čistilec

Carver-Cleaner ali Izklesovalec-Čistilec je modul ki skrbi, da se datotečni sistem ne pokvari ob brisanju že izbrisanih datotek. Modul ni potreben če uporabnik potrebuje le selektivno brisanje tega kar je na disku. Če pa želimo sistem ki sovpada z členom 100a Nemškega zakonika o kriminalistični preiskavi, ki določa da moramo osebne podatke ki se ne navezujejo na primer izbrisati, potem moramo sčistiti tudi podatke ki niso direktno dostopni. Do teh podatkov pridemo preko izklesovanja. Problem pa lahko nastane če želimo izbrisati podatke, ki sovpadajo s trenutnim datotečnim sistemom. Torej če že samo en blok datoteke, ki jo

želimo izbrisati, sovpada z blokom datoteke ki je navedena v $MFT, bi izbris pomenil okvaro datotečnega sistema. Zato je namen tega modula da preveri če se slučajno datoteke, ali bolje bloki, ki jih želimo izbrisati iz diska slučajno prekrivajo z bloki shranjeni v $MFT. Tako so nato označeni za izbris le bloki, ki se ne prekrivajo.

4.2.4 Hashkalkulator

Hashkalkulator (Hashcalculator), je modul ki izračuna varnostno vsoto celotnega sistema ter skrbi da se ne pokvari integriteta dokazov. Za izračun je uporabljeno Merklovo drevo, kateremu lahko uporabnik definira velikost bloka, in število otrok na vozlišče. Za izračun varnostne vsote se uporablja protokol MD5. Izračun drevesa poteka tako, da se nad bloki (katerim velikost je definiral uporabnik) izračuna varnostna vsota, nato pa se uporabniško določeno število varnostnih vsot združi v skupno varnostno vsoto. Ta korak se ponavlja dokler nam ne ostane le ena skupna glavna varnostna vsota.

Izračun višine drevesa h se izračuna z enačbo 1. Ker pa želimo imeti uravnoteženo drevo moramo izračunat število listov ter iz tega nato število ničelnih blokov ki jih moramo dodati drevesu. Število listov se izračuna z enačbo 2.

$$h = \lceil \frac{\log_{10} \frac{velikost\_diska}{velikost\_bloka}}{\log_{10} st\_otrok\_na\_vozlisce} \rceil \quad (1)$$

$$st\_listov = (st\_otrok\_na\_vozlisce)^h \quad (2)$$

Merklovo drevo izračunamo pred kakršnim koli spreminjanjem slike diska ter nato še po brisanju. Drevesi sta shranjeni v seznam XML datotek. Ta dva drevesa primerjamo, da vidimo kateri deli diska so bili spremenjeni. S tem zagotovimo, da se niso izbrisali ali spremenili nobeni podatki, kateri niso bili označeni za izbris in bi lahko bili relevantni za sodišče.

4.2.5 Izbrisovalnik

Centralni modul vtičnika je Izbrisovalnik ali Deletion Module. V njem je implementiran jedrni del programa. To je brisanje podatkov iz diska ter urejanje metapodatkov v $MFT. Modul sestavljata dva dela. Prvi je namenjen brisanju podatkov, kar naredi tako da bloke namenjene brisanju nastavi na 0. Drugi del je namenjen ugotavljanju ali se lahko podatke v $MFT enostavno pobriše ali pa se jih mora urediti, da z brisanjem ne bi pokvarili datotečnega sistema. Slednji primer se na primer zgodi ko je potrebno urediti strukturo B-drevesa.

Pred urejanjem slike diska je potrebno pridobiti pravice za pisanje, česar Digital Forensic Framework ne dovoljuje, slike pa so vedno priklopljene na orodje samo v bralnem načinu. Tako je potrebno z rutinami DFF pridobiti pot do diska. Ko pridobimo pot, lahko nato z C++ rutinami odpremo sliko s pravicami branja in pisanja. Pomembno je tudi da sinhroniziramo kazalce, saj bi drugače lahko prišlo do brisanja napačnih delov diska.

Algoritem deluje rekurzivno. Začetna točka je korenska mapa, nato pa rekurzivno vstopa v podmape ter vsako datoteko ali podmapo, ki je označena za brisanje dodatno pregleda. Vsi bloki podatkov ter metapodatkov označeni za brisanje so shranjeni v notranji spomin programa. Po pregledu celotne mape se preveri ali so bili katerikoli podatki ali podmape v njej spremenjene. Če je temu tako, potem to pomeni da se mora spremeniti tudi B-drevo saj bi drugače datotečni sistem javil napako.

Spreminjanje B-drevesa je zahtevno ter kritično opravilo saj nam lahko napačna sprememba drevesa prepreči nadaljni dostop do datotek, ki niso bile izbrisane. Urejanje drevesa je odvisno od tega ali je brisani vnos na listnem ali vozliščnem nivoju.

Če je na listnem nivoju, potem si je potrebno zapomniti velikost vnosa. Nato prepišemo brisani vnos z vnosi ki so za njim v listu. Potem moramo za zadnjim vnosom ki smo ga prepisali zapisati ničle do velikosti brisanega vnosa. Če je vnos na vozliščnem nivoju, potem se rekurzivno pregleda del drevesa levo od brisanega vnosa in poišče najvišjo vrednost vnosa. Ta se shrani ter začasno izbriše iz drevesa, kot je opisano na začetku tega odstavka. Odvisno od razlike velikosti shranjenega vnosa in brisanega vnosa, shranjen vnos prepiše brisani vnos. Če s tem poddrevo postane zastarelo, se pobriše tudi zastavica ki označuje otroka.

Ker so velikosti gruče v NTFS specifičnih velikosti se ob prepisovanju lahko dogajajo prelivi, zato je potrebno posodabljati vrednosti o uporabljeni velikosti v gruči. Prav tako je potrebno posodabljati $fixup$ vrednosti, s katerimi se preverja koherentnost podatkovnih blokov.

Brisanje podatkov, metapodatkov in urejanje B-drevesa so kritične operacije, katere želimo izvesti istočasno in v čim manjšem časovnem oknu. Ker se lahko zaradi raznih okoljskih razlogov program nepričakovano zaustavi so avtorji implementirali Izbrisovalnik tako, da se pred kakršno koli modifikacijo datoteke ter njenih metapodatkov vse modifikacije zabeležijo. Šele ko se celotni pregled diska zaključi ter se shranijo vse potrebne modifikacije se te modifikacije izvedejo. S tem se zmanjša kritično okno, saj ob izvajanju modifikacij ni pregledov kaj je še potrebno urediti, ali bolje, so bili te že izvedeni.

Izbrisovalnik tudi zabeleži vse modifikacije datotek v beležko. Beleženje je zelo podrobno saj se zabeležijo podatkovni bloki datoteke ter pot do nje. Tukaj se tudi pojavi vprašanje ali tako beleženje krši člen 100a nemškega zakona o kazenskem postopku, saj lahko iz takih beležk razberemo osebne podatke.

5. NADALJNJI RAZVOJ

Na področju še vedno manjkajo implementacije selektivnega brisanja za vse datotečne sisteme in ki bi obenem še ustrezale forenzičnim standardom. Verjetno bo minilo še nekaj časa preden se bodo tovrstna orodja zares uveljavila med digitalnimi forenziki.

Možnosti za nadaljni razvoj na širšem področju je tudi v sistemih za inteligentno prepoznavanje relevantnih podatkov[6]. Sistemi, ki akumulirajo znanje forenzikov, bi lahko asistirali pri iskanju relevantnih podatkov ali pa jih celo sami

avtomatsko zajeli. Vendar pa je pot, preden bodo taki sistemi zares uporabni, še zelo dolga.

6. VIRI

[1] K. M. Alex Chumbley and J. Khim. Merkle tree. https://brilliant.org/wiki/merkle-tree/. Dostopano: 5.6.2018.

[2] A. Castiglione, G. Cattaneo, G. D. Maio, and A. D. Santis. Automatic, selective and secure deletion of digital evidence. In *2011 International Conference on Broadband and Wireless Computing, Communication and Applications*, pages 392–398, Oct 2011.

[3] Evropska konvencija o varstvu človekovih pravic in temeljnih svoboščin. http://www.varuh-rs.si/pravni-okvir-in-pristojnosti/mednarodni-pravni-akti-s-podrocja-clovekovih-pravic/svet-evrope/evropska-konvencija-o-varstvu-clovekovih-pravic-in-temeljnih-svoboscin/. Dostopano: 5.6.2018.

[4] J. Stüttgen, A. Dewald, and F. C. Freiling. Selective imaging revisited. In *2013 Seventh International Conference on IT Security Incident Management and IT Forensics*, pages 45–58, March 2013.

[5] The german code of criminal procedure stpo. https://www.gesetze-im-internet.de/englisch_stpo/englisch_stpo.html#p0488. Dostopano: 5.6.2018.

[6] P. Turner. Selective and intelligent imaging using digital evidence bags. *Digital Investigation*, 3:59 – 64, 2006. The Proceedings of the 6th Annual Digital Forensic Research Workshop (DFRWS '06).

[7] Zakon o kazenskem postopku uradno prečiščeno besedilo (zkp-upb8). https://www.uradni-list.si/1/objava.jsp?sop=2012-01-1405. Dostopano: 5.6.2018.

[8] C. Zoubek and K. Sack. Selective deletion of non-relevant data. *Digital Investigation*, 20:S92 – S98, 2017. DFRWS 2017 Europe.

Tools and methods for falsification of SMS messages

Ladislav Škufca
Faculty of Computer and Information Science
University of Ljubljana
ls8856@student.uni-lj.si

Luka Podgoršek
Faculty of Computer and Information Science
University of Ljubljana
lp5796@student.uni-lj.si

Agáta Ďarbujanová
Faculty of Informatics
Masaryk University
xdarbuj@mail.muni.cz

ABSTRACT

In this paper we present problem of presenting SMS messages in court as a digital evidence. SMS messages are still one of the most used forms of communication, therefore exists possibility to be present in court as evidence. This article consists of several parts. First, we describe how SMS messages work regardless of platform and the usage of public SMS gateways. Then we describe how SMS messages were falsified in the past (around 2010) on Nokia phone. Lastly, we present contemporary processing on modern devices like Android and iOS with an experiment of falsification of SMS on Android device.

Keywords

SMS message, mobile forensics, falsification of SMS, SMS gateways, digital evidence, bulk SMS, SMS on Android

1. INTRODUCTION

One of many forms of electronic communication technologies are SMS (Short Message Service) messages. SMS messages were very popular in the past and are still widely used today even though new ways of communication have been developed. Beside SMS messages people also communicate with Instant Messaging tools like WhatsApp, Viber, Messenger etc. Today SMS messages are still being used for communication but also for marketing or as a security element of the authentication service. Overall SMS messages are very useful tool for communication but they also bring some risks because they could be easily falsified or sent anonymously[5]. Today we have many different ways of sending them, because they have a lot of potential for enterprises. Some of these enterprises need to receive a lot of SMS messages from their customers (e. g. charity or voting in the competition), other need to send a large amount of SMS messages (e. g. confirmation code for logging to Internet banking or just advertisement news). SMS messages are powerful tool but can also be used by criminals. Therefore, it is important to understand how SMS messages work, if they are trustworthy and suitable for digital forensics investigations.

1.1 MOTIVATION

The topic about falsifying SMS messages arises a lot of questions but the most important one is: *Can obtained SMS messages be trusted and used in court as a valid evidence?* The question is quite interesting from the aspect of this course and it was the main reason for the three of us to choose this topic.

2. HOW SMS WORKS

Messages are sent to a Short Message Service Center (SMSC), which provides a "store and forward" mechanism[17]. SMSC is a network element in the mobile telephone network. Its purpose is to store, forward, convert and deliver SMS messages[12]. If a user sends a text message to another user, SMSC serves as a gateway to store and forward this message to the recipient when available[18]. SMS message is stored only temporary on an SMS center. Expiration date can be set on most mobile handsets. After that time, the SMS message should be deleted and thus no longer available for dispatch to the recipient's mobile phone. It turns out that the SMSC itself can be configured to ignore or differently handle the delivery of the message. SMS message also supports status reporting about the delivery of itself. This functionality can be turned on by visiting mobile handset's settings.

The SMS is currently defined in 3GPP as TS 23.040, latest version 15.0.0 from 27.03.2018 [16]. The Short Message Service contains 8 main elements: ValidityPeriod, ServiceCentreTimeStamp, ProtocolIdentifier, MoreMessagestoSend, Priority, MessagesWaiting, AlertSC and MT Correlation ID. From forensics perspective the most important part is the ServiceCentreTimeStap. It is the information element by which the service center informs the recipient mobile station about the time of arrival of the short message. The short message format is defined in RFC 5724[13]. The SMS message consists of: recipient phone number (or multiple of them with additional tag), body and optional tags for additional info - for example tag for multiple SMS messages, etc. The most important part of this format is that there is no date or time presented (timestamp is marked on gateways, phone itself, etc). It is also important to know that the body of the format has a maximum length of 160 characters. SMS messages can be sent over GSTN (General switched tele-

phone network) or some Web-based services, which are not directly connected to a GSTN network. The SMS format RFC also provides a short security recap for messages, saying: SMS messages are transported without any provisions for privacy or integrity, so SMS users should be aware of these inherent security problems of SMS messages. Unlike electronic mail, where additional mechanisms exist to layer security features on top of the basic infrastructure, there currently is no such framework for SMS messages[13].

In the past it was possible to receive SMS just to mobile phone device or to ESMEs (External Short Messages Entities[1]), however with the advent of smart phones, new opportunities have opened up. For example "messages that are delivered to a mobile device may not remain restricted to that device"[10], because user can use some synchronization via cloud services with some other applications (e. g. Facebook Messenger or Google). This means new opportunities for attackers to change SMS meta data or content. Another security risk could be in the way of processing an SMS, because it is processed by "many different entities", so anyone can change it.

It is not obligatory to send SMS message only from one mobile phone device to another. There exist some public gateways for receiving, sending bulk SMS and even sending SMS messages for free [2]. The first group of gateways, receiving SMS, can be useful when a criminal does not want to use his or her personal number. A felon could go to some gateway website and use a public number for receiving messages [10]. An received SMS is then visible on the web site and there is no opportunity to get information about real receiver. Investigators can ask service provider for log records, but because this service is public and has a lot of different users, they could get a lot of suspects.

The second group is sending bulk SMS messages. This is often used by commercial companies like outsourcing service. They often offer transactional, promotional and business/enterprise types of SMS messages. Transactional means sending OTPs and alerts to registered users, promotional refers to some offers and business is just a marketing term for combination of promotional and transactional types of messages [1]. This kind of SMS-messaging, also known as application-to-peer messaging (A2P Messaging) or 2-way SMS, continue to grow steadily at a rate of 4% annually[17]. Businesses that use these services can also choose how to fill a database of recipients or how to create content of the SMS messages. Usually they can use web application, smart phone application, export from excel file, export from e-mail, etc. A lot of these companies are in India [3] Bulk SMS companies promise their customers that "any recipient of any message has the right to know the identity of the sender, and this will be disclosed on request to the recipient" [11]. Interesting fact is that Bulk SMS does not require a name of customer's company on their registration form [4]. We tried another bulk SMS services providers and it was often possible to write some random characters in submission form. Usually these forms only verified that we are people and not robots. From this we suppose that everybody can buy this kind of service.

There are many gateway offers on internet but here are listed only 3 of them (they offer sending of SMS in Slovenia): `https://www.smsgateway.to/en/slovenia`, `https://www.clickatell.com/` and `https://www.smsapi.si/`. Average price from these services is around 0.03 EUR per SMS message - but the price may vary. It usually depends on the number of messages you want to send per month and how you are willing to pay for reliability. There is even an application on Google Play which turns your own Android phone into a SMS gateway `https://smsgateway.me/`.

It is also very important to say that there must be active TMSI (The Temporary Mobile Subscriber Identity) [8] if we want for a SMS message to be sent/received successfully. The aim of TMSI - mobility management is to track where the subscribers are so they can be reachable for all kinds of mobile services, for example an SMS.

3. RECAP OF THE ARTICLE

This section is a recap of the article about falsifying SMS Messages[7] that was a basepoint for our research.

One of the first research, which suspected that commonly available methods and tools for digital mobile investigation (e. g. HEX dump[5]) can be misused for falsification or modification of SMS messages, is from 2010. There are some areas how to influence the digital evidence during investigation. The first one is that examination tools could generate an inaccurate report, the second one is that the report could be unjustifiably modified and the last one is that SMS messages could be modified on the device. The last case is analyzed in above-mentioned article and this paper.

Researchers tried to falsify or modify the service center address, a content of the message, sender's address, data and time stamp. They choose commercial tools which cannot be modified (no open source software) because of reliability of their experiment. All experiments were made on Nokia 6021 with the tool Sarasof UFS/HWK (also called Tornado). This tool was chosen because it covered the most of capabilities of concurrent software. The original aim of these instruments is not to falsify an SMS like digital evidence, but it should be used for upgrading software by manufacturers.

This model of Nokia mobile phone was chosen for many reasons. Nokia was the most widely manufactured mobile phone in that time. It was well structured and it used PM tables. PM tables contain keys which represent category and subkeys for referencing related data. On this model of mo-

[1]This can provide SMS message services as donation to charities, one time password or emergency alerts[10]

[2]These are usually anonymous SMSes, because it is not possible to add an sender's phone number. However, usually is added something like advertisement of the sending company, co investigators can ask the company for log file.

[3]E. g. TextLocal: `https://www.textlocal.in/` or Satej Infotech: `http://www.satejinfotech.in/`.

[4]We tried Bulk SMS: `https://www2.bulksms.com/register/`, Text Magic: `https://my.textmagic.com/register/`, Click Send: `https://dashboard.clicksend.com/#/signup/step2/input-number`

[5]HEX dump is "a form of mobile telephone examination which can recover deleted and/or hidden system information".

bile phone manufacturers did not keep the data format accordingly to GSM 3.40 standard, which is technical realization of SMS messages. They "reordered the same elements and included its own parameters/fields". Date and time stamp are stored in reversed decimal values. For SMS content was used format with length fields (User Data Length[6], length of encoded Protocol data unit (PDU) [7] content and encoded message structure length). Message length has to be 8 octets, so each message has been added a suffix 0x55 if the length of the message was shorter. Timestamp of multipart SMS is set to the last part of the message.

Two kinds of test were done in the research mentioned above. The first one was with expected (valid) input values and the second one with invalid values. Modification of the service address center, sender's address and content of the SMS text messages could be edited if relevant length values "were updated to reflect the new length of the data". Time and date stamp and time zone could be modified without problems and this change could be detected only in the PDU. Researchers did not find a way to detect that data was changed, because the new values were visible on the mobile hand set.

The test with invalid input values was done only on timestamps, because here it should be used some entry conditions. Some input restrictions are here (day can get value from 1-31 etc.), but is not well implemented. E. g. it does not check the number of days in month. It was also possible to enter time value which does not make sense (e. g. timestamp 13:99:99). This invalid timestamps were interpreted in different ways by different tools. Some of them show it without modification, but the others show it as valid time (e. g. instead of "99" it was "42"). However, examiners were not able to create a new false message on the device, they were only able to edit existing SMS "without access to privileged hardware and software". The new false SMS message could probably be created on another device and then sent to the mobile phone.

There exist some ways to detect if the message was changed or not. Data about SMS messages in a mobile phone device should coincide with the information from the mobile service provider. It could be problematic to identify changed content of the SMS message, because the mobile service provider does not keep content of the message. It is also necessary to check the device date and time settings because it can influence the timestamp.

4. SMS STORAGE LOCATION ON MODERN DEVICES

Unlike Nokia 6021 which stores SMS messages in PM tables, modern mobile devices store SMS and MMS messages in specialized files. These files are SQLite databases.

SQLite database is open-sourced and most widely deployed database in the world. It does not need separate server process to manage database therefore it is perfect for usage in mobile devices. Because everything is stored in one file, SMS and MMS messages can be easily copied, backed up or uploaded to a cloud service.

We have looked into Android and iOS devices to find where these files are located. Both Android and iOS have specific location for these files.

4.1 Android devices

Android is currently one of the most widely used mobile phone operating system like above mentioned Nokia in it's time. Different messaging applications can store SMS and MMS messages in different locations but they store it in databases. Therefore we know what to look for. On Android devices SMS and MMS messages are usually stored in **/data/data/com.android.providers.telephony /databases/mmssms.db**.

If we do not know the location of **mmssms.db** file we can use Android *adb* tool. With this tool we can connect shell to Android device.

Some useful ADB commands:

- adb devices - list connected devices
- adb shell - runs interactive shell
- adb root - connects with root to device (we need rooted device)
- adb push - uploads file to device
- adb pull - downloads file from device

If we run *adb shell* we will get interactive shell connected to our Android device. Then we can run $ *find / -name "*mmssms*"*[14] to search for SMS and MMS messages database. In order to get result from command above we need **root** access to device. Bellow are command which you can use to get location of SMS and MMS database.

```
// find connected devices
$ adb devices
// connect shell to device
$ adb shell
// search for sms database
$ find / -name "*mmssms*"
```

If we have root access to device then we can easily search for SMS and MMS database. Once we find it we can copy it and view it with any application for SQLite databases [14, 15].

4.2 iOS devices

Like Android devices iOS devices store SMS and MMS messages in SQLite database. Throught different versions of iPhones file was always named **sms.db**. If we have a backup of our phone made through iTunes we can find it on macOS computer in `~/Library/Application Support/MobileSync/Backup/` or we can retrieve this file through Apple iCloud backup.

[6]User Data Length represents "number of characters in the user's message" [7].
[7]PDU is information which "is transferred among peer entities of a network, such as a communications network or a computer network" [19]

File name will be encoded with SHA-1 hash function. Example of typical file name: 3d0d7e5fb2ce288813306e4d4636395e047a3d28 [9].

With the use of SQLite database viewer application we can get access to the messages. In theory, it would be possible to modify the data with SQLite database editor and restore the modified backup back to the phone.

5. EDITING SMS CONTENT ON ANDROID DEVICE

If we want to edit SMS content on Android device we need to have root access on device. In order to get root access on device we need to root it. In our research we have used Android device *Samsung S3 mini*. Before we could extract *mmssms.db* file we have rooted mobile device. If you do not know how to root device you can follow this guide[4].

Once we had root access on mobile device we connected it to the computer. Then we used Android *adb* tool. With this tool we connected to the device as root and searched for mmssms.db file. File was located in /data/data/ com.android.providers.telephony/databases/mmssms.db.

Because data in /data/data directory is protected we cannot directly copy it onto computer. We found a workaround around this problem[3]. First we copied mmssms.db file to external SD card.

```
adb shell
$ mkdir /mnt/extSdCard/tmp
# su
# cat /data/data/com.android.providers.telephony/
    ↪ databases/mmssms.db > /mnt/extSdCard/tmp/
    ↪ mmssms.db
# exit
$ exit
```

Afterwards we copied file to computer with *adb*. Command bellow will copy SMS database to your working directory.

```
adb pull /storage/extSdCard/tmp/mmssms.db .
```

When we had copied the file we opened it with *DB Browser for SQLite*. Afterwards we searched for **sms table**. In there each row represented SMS message. Under column **body** we found SMS contents for each SMS message. Now we had complete control over SMS contents. We changed content of two SMS messages and saved changes. Now we needed to upload changed content back to Android device.

We could not upload changed database directly to /data/-data/com.android.providers.telephony/databases/mmssms.db. First we needed to upload changed database to external SD card.

```
adb push mmssms.db /data/tmp/mmssms.db
adb shell
$ su
# cd /data/tmp
# mv mmssms.db /data/data/com.android.providers.
    ↪ telephony/databases/mmssms.db
# exit
$ exit
```

Afterwards we have rebooted mobile device and checked if falsified SMS message is visible on mobile device. You can see our result on images bellow.

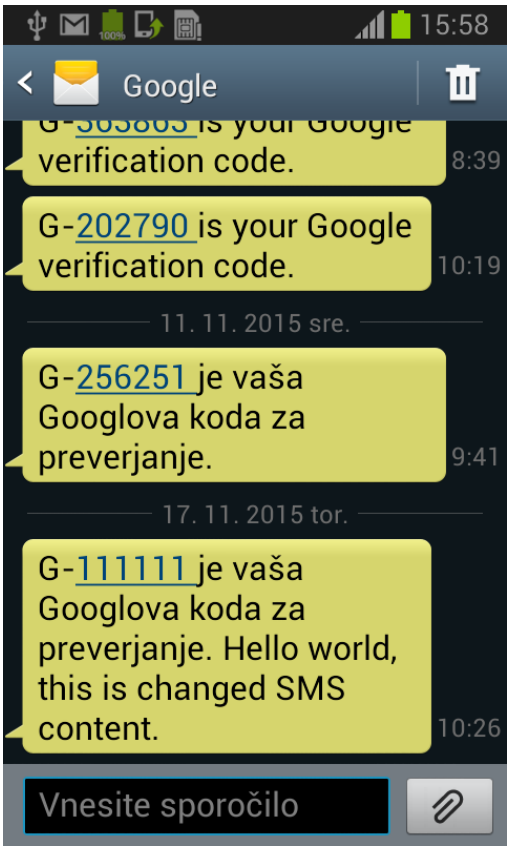


Figure 1: Edited SMS content of original SMS from Google

We managed to upload and view falsified SMS message on rooted Samsung S3 mini. As you can see only content was changed. Date and time stayed the same as it was before we tried to modify and SMS data.

Last step would be to remove root access on a mobile device. This way an inspector probably would not notice that anyone has tampered with mobile device. Because we only wanted to show that you can falsify SMS message on Android device we have skipped this step.

6. SMS AS DIGITAL EVIDENCE

Sending SMS messages can provide feeling of anonymity, because a phone number does not have to be directly connected with the owner. It could be one reason why it is used by drug dealers and other criminals for their illegal activity. However, it is necessary to know how important role can SMS play in the court.

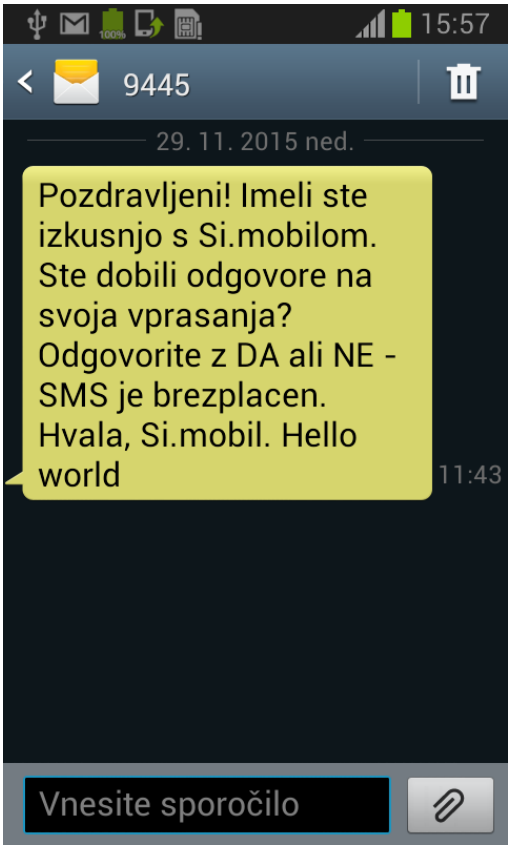


Figure 2: Edited SMS content of original SMS from Simobil

Criminals can use different techniques to confuse investigators. They can falsify contents, timestamps and sender of the SMS message, use anonymous SMS or use a lot of phone numbers for their operation. That is why is important to mention the way how to detect an author of the message. This problematic was interesting for Ishihara [6]. A situation could be like this: investigators get SMS messages from criminals [8] and another from suspects. In [6] was developed and tested a way how to recognize if some set of SMS is from the same author. It is possible to determine it, because every person has original writing style, which includes using of emojis, typography, grammar, favorite words, etc. Ishihara [6] was focused on using spaces, punctuation and used words. He agreed with Mohan [2] that is possible to correctly predict author of an SMS message, the first above mentioned researcher with accuracy of 80 % and the second one with 65%∼70%. Unfortunately, a big sample of messages is required.

7. CONCLUSION

We have demonstrated how to change SMS message content on a mobile device. Next question that we asked ourselves was how to check if SMS content was changed. In order to verify integrity of SMS message we would need the phone of another person who sent/received the SMS message. If content is the same on both devices then we can assume that the message was not altered. If the contents is different, than we know that somebody has tampered with the content. Other way would be to check with telecommunication providers and verify SMS content but it may vary between providers if they do log SMS message data. The answer to this may be obtainable from the provider itself, probably from provider's terms of use.

However, providers usually store only sender, receiver, time and date information, because the whole SMS messages are deleted after sending them from SMSC to the receiver[5]. If the content of the SMS is still stored, we can detect the changed SMS message data, otherwise we wouldn't be sure if the SMS message was really the same.

SMS messages can be used in court as an evidence in Slovenia if they we were collected according to law. As we have seen in our research, someone could easily tamper with such messages - therefore we need forensics expert to verify integrity of such messages. It would also be interesting to see what data exactly do Slovenian telecommunication providers store.

[8]Investigators are sure that it is from criminals, e. g. SMS about sexual harassment.

8. REFERENCES

[1] Text local. Available: `https://www.textlocal.in/sending-bulk-sms`, 2018. [Accessible: 12. 05. 2018].

[2] M. K. R. A. Mohan, I. M. Baggili. Authorship attribution of sms messages using an n-grams approach. *CERIAS Tech Report*, (11), 2011.

[3] backup full sms/mms contents via adb. Available: `https://stackoverflow.com/questions/12266374/backup-full-sms-mms-contents-via-adb`, 2018. [Accessible: 11. 05. 2018].

[4] [guide][xxamg1][4.12] root for galaxy s iii mini i8190. Available: `https://forum.xda-developers.com/showthread.php?t=2030282`, 2018. [Accessible: 11. 05. 2018].

[5] L. R. C. Graeme Horsman. An investigation of anonymous and spoof sms resources used for the purposes of cyberstalking. *Digital Investigation*, (13):80–93, 2015.

[6] S. Ishihara. A forensic authorship classification in sms messages: A likelihood ratio based approach using n-gram. *Proceedings of Australasian Language Technology Association Workshop*, pages 46–56, 2011.

[7] C. Marryat. Falsifying sms messages. *SSDDFJ*, 4(1), September 2010.

[8] Mobility management. Available: `https://en.wikipedia.org/wiki/Mobility_management`, 2018. [Accessible: 11. 05. 2018].

[9] How to access and read the iphone sms text message backup files. Available: `http://osxdaily.com/2010/07/08/read-iphone-sms-backup/`, 2018. [Accessible: 11. 05. 2018].

[10] B. Reaves. Sending out an sms: Characterizing the security of the sms ecosystem with public gateways. *Security and Privacy*, pages 339–356, 2016.

[11] Bulk sms privacy policy. Available:

https://www.bulksms.com/company/data-protection-and-privacy-policy.htm, 2018. [Accessible: 12. 05. 2018].

[12] Short message service center. Available: https://en.wikipedia.org/wiki/Short_Message_service_center, 2018. [Accessible: 11. 05. 2018].

[13] Short message service (sms) message format. Available: https://www.loc.gov/preservation/digital/formats/fdd/fdd000431.shtml, 2018. [Accessible: 11. 05. 2018].

[14] Where on the file system are sms messages stored? Available: https://android.stackexchange.com/questions/16915/where-on-the-file-system-are-sms-messages-stored, 2018. [Accessible: 11. 05. 2018].

[15] Raw access to sms/mms database on android phones. Available: http://www.toughdev.com/content/2012/02/raw-access-to-smsmms-database-on-android-phones/, 2018. [Accessible: 11. 05. 2018].

[16] Technical realization of the short message service (sms). Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=747, 2018. [Accessible: 11. 05. 2018].

[17] Sms. Available: https://en.wikipedia.org/wiki/SMS, 2018. [Accessible: 11. 05. 2018].

[18] Store and forward. Available: https://en.wikipedia.org/wiki/Store_and_forward, 2018. [Accessible: 11. 05. 2018].

[19] M. H. Weik. *Computer Science and Communications Dictionary*. Springer, Boston, 2000.

Hash tehnike za forenzične preiskave mobilnih naprav

Seminarska naloga pri predmetu Računalniška forenzika

Marko Ambrožič
63090095

Rok Šeme
63110323

POVZETEK

Raziskave, opravljene na National Institute of Standards and Technology, so pokazale, da so hash vrednosti notranjih pomnilnikov mobilnih naprav spremenljive pri opravljanju back-to-back pregleda. Hash vrednosti so koristne pri zagotavljanju izvedencem možnosti filtriranja znanih podatkovnih datotek, ujemanja podatkovnih objektov na različnih platformah in dokazati, da integriteta podatkov ostaja nedotaknjena. Raziskava, izvedena na Univerzi Purdue, je primerjala znane hash vrednosti z izračunanimi vrednostmi za podatkovne predmete naložene na mobilne naprave z različnimi metodami prenosa podatkov. Medtem ko so bili rezultati za večino preizkusov enotni, so bile hash vrednosti, ki so bile izračunane za podatkovne predmete, prenesene prek storitve za večpredstavnostna sporočila (MMS), spremenljive.

Ključne besede

Forenzična preiskava mobilnega telefona, Forenzična preiskava mobilne naprave, Hash, MMS, MD5

1. UVOD

Hitra porast uporabe mobilnih naprav v zadnjih letih je odprla veliko novih možnosti za raziskave in analize v digitalni forenziki. Mobilne naprave so postale stalni partner vsakega človeka in tako zbirajo ter hranijo praktično vse podatke o našem življenju - kje smo se gibali, kaj smo počeli, itd. Poleg iskanja načinov za analizo teh izjemnih količin podatkov pa se pri mobilni forenziki pojavlja tudi težava, kako zagotoviti istovetnost podatkov, ki so prenešeni preko mobilnega omrežja in drugih načinov brezžične komunikacije. Tu imamo v mislih predvsem sms in mms sporočila. Običajna praksa v digitalni forenziki je ta, da se istovetnost podatkov zagotovi z izračunom vrednosti zgoščevalne funkcije (npr. SHA-1 ali MD5). V tem delu bomo preverili kako se ta pristop obnese pri forenziki mobilnih naprav in kakšne težave lahko pri tem nastanejo. Na koncu bomo poskušali podati usmeritev za nadaljne raziskave.

1.1 MD5

Algoritem MD5[1] je široko uporabljana zgoščevalna funkcija. Deluje tako, da iz nekega besedila poljubne dolžine izračuna 128-bitno številko. Zamišljen je bil kot kriptografska zgoščevalna funkcija. Zaradi velikega števila pomanjkljivosti v kriptografiji ni več uporabna. Še vedno pa lahko služi kot kontrolna vsota proti nenamerni spremembi podatkov.

1.2 SHA-1

Algoritem SHA-1[2] je kriptografska zgoščevalna funkcija, ki vzame besedilo poljubne dolžine in izračuna 160-bitno številko. Algoritem je zasnovala ameriška vladna agencija NSA. Od leta 2005 algoritem ni več varen proti napadalcem z dovolj finančnimi sredstvi. SHA-1 je težje razbiti kot MD5 in lahko služi kot kontrolna vsota pri preverjanju integritete podatkov.

2. SORODNA DELA

Članek z naslovom Fighting Crime With Cellphones' Clues[6] jedernato opisuje pomembnost mobilnih naprav in podatkov na le-teh za forenzično preiskavo. Digitalni dokazi pridobljeni na mobilnih napravah lahko pomenijo razliko med težkim primerom brez konkretnih dokazov in primerom kjer so dokazi na dlani in tako olajšajo postopek na sodišču. Zaradi tega je toliko bolj pomembno, da je avtentičnost dokazov neizpodbitna. Rick Ayers z inštituta NIST v delu Guidelines on Mobile Device Forensics (Draft) [3] podaja smernice za forenzično analizo mobilnih naprav kjer večkrat poudarja, da je izračunavanje vrednosti zgoščevalnih funkcij z namenom zagotavljanja integritete podatkov, najpomembnejša lastnost katerega koli forenzičnega orodja. Ta vrednost mora ostati enaka skozi celotno življenjsko obdobje vsake datoteke uporabljene kot dokazno gradivo. Zagotavljanje integritete podatkov ne daje le teže dokazom ampak omogoča, da morebitne nadaljnje analize pridejo do istih ugotovitev. Poleg pomembnosti izračuna vrednosti zgoščevalnih funkcij poudarja tudi, da so mobilne naprave ves čas aktivne in v neki meri stalno spreminjajo podatke na napravi. Vrednosti zgoščevalnih funkcij bodo pogosto ob dveh zaporednih zajetjih vseh podatkov različne. To težavo lahko zaobidemo z izračunom vrednosti za vsako datoteko ali imenik posebej. Rizwan Ahmend in Rajiv V. Dharaskar se v svojem delu Study of cryptographic hashing: An analysis from the perspective of developing efficient generalized forensics framework for the mobile devices [5] ukvarjata s postavljanjem splošnega ogrodja za izvajanje forenzične analize mobilnih naprav. Bolj podrobno predstavita najbolj pogosti zgoščevalni funkciji MD5 in SHA-1 ter njune ranljivosti. Glavne

lastnosti, ki jih mora imeti idealna kriptografska zgoščevalna povzameta v naslednjih štirih točkah:

- *Enostavnost izračuna vrednosti za kateri koli podano sporočilo.*
- *Izračun originalnega sporočila iz vrednosti funkcije je neizvedljiv.*
- *Sprememba sporočila brez spremembe vrednosti funkcije je neizvedljiva.*
- *Neizvedljivo je najti dve različni sporočili z isto vrednostjo zgoščevalne funkcije.*

Na podlagi teh lastnosti nato predstavita pomankljivosti MD5 in SHA-1. Izpostavita delo Xiaoyun W. in Hongbo Y. [8], kjer je bilo pokazano, da je mogoče v praksi zgraditi dve sporočili, ki vrneta isto vrednost z uporabo zgoščevalne funkcije MD5. Možnost razbitja funkcije SHA-1 je bilo teoretično dokazano a je v praksi zelo težko izvedljivo. Zaradi izredne nepraktičnosti napadov na funkciji MD5 in SHA-1, sta v forenziki še vedno priznani in široko uporabljani. Avtorja ugotavljata, da sta obe funkciji tudi zanesljivi pri izračunu vrednosti za posamezne datoteke na mobilnih napravah, izpostavljata pa opaženo nekonsistentnost pri izračunu vrednosti za slike prenešene z uporabo mms sporočil.

3. TERMINOLOGIJA

- *Metode prenosa podatkov*: Komunikacijski kanali (npr. Bluetooth, MMS, itd.), ki zagotavljajo prenos, s katerim se napolni notranji pomnilnik mobilnih naprav.
- *Varnostni hash*: Matematični algoritem, ki vzame poljuben blok podatkov in vrne bitni niz fiksne velikosti, hash vrednost, tako da bo vsaka sprememba podatkov spremenila vrednost razpršitve.
- *Podatkovni objekti mobilne naprave*: Posamezne datoteke (npr.: .jpg, .bmp, .gif, itd.), ki se nahajajo v notranjem pomnilniku mobilne naprave.
- *Forenzično orodje za mobilne naprave*: Orodja namenjena za izvajanje logičnega pridobivanja podatkov iz notranjega pomnilnika mobilnih naprav.
- *Forenzično orodje za osebne računalnike*: Forenzična orodja, namenjena pridobivanju podatkov s trdih diskov (npr.: IDE, SATA, SCSI, itd.)

4. METODOLOGIJA

Začetna priprava se je začela z izračunavanjem MD5 hash vrednosti za posamezne podatkovne datoteke, prikazane v sliki 1. Vsaka posamezna grafična datoteka je bila prenesena na forenzično delovno postajo in se izračunal MD5 hash z orodjem *Access Data's Forensic Toolkit*. Orodje je bilo izbrano glede na dostopnost. Hash vrednosti, ki so jih sporočili za pridobljene podatkovne objekte s forenzičnimi orodji za mobilne naprave, so bile primerjane z znano začetno vrednostjo.

Mobilne naprave so bile izbrane izključno na podlagi razpoložljivosti in podobnih nastavitev (npr.: MMS, Bluetooth, notranja kamera). Izbrali so osem duplikatov mobilnih naprav (tj. izdelava, model, strojna programska oprema). Z uporabo duplikatov mobilnih naprav je mogoče ugotoviti ali forenzična orodja za mobilne naprave poročajo o skladnih hash vrednostih za vnaprej določene podatkovne objekte v skupnih mobilnih napravah. Dvoje forenzičnih orodij za mobilne naprave (Paraben's Device Seizure [4] in Susteen's Secure View [7]) sta bili izbrani zaradi razpoložljivosti, vgrajene hash funkcije in podpore pridobitve za izbrane mobilne naprave.

Obstajajo številni načini prenosa podatkov na mobilno napravo. Izvedenih je bilo več testov, da se ugotovi ali hash vrednosti ostajajo skladne v različnih metodah prenosa podatkov. Uporabljene so bile naslednje metode prenosa podatkov: universal memory exchanger, MMS, Bluetooth in MicroSD. Izvedeni so bili dodatni orientacijski testi, da bi ugotovili, ali so bile hash vrednosti spremenjene pri: a) spreminjanju vloge shranjene grafične datoteke (npr. shranjevanje kot ozadje) in b) prenosu posnetka posnetega z notranjo kamero mobilne naprave na forenzično delovno postajo.

Raziskava je vključevala več ciljev, ki so bili: a) ugotovitev, ali se pojavijo neskladja med znanimi hash vrednostmi, b) dokumentirati, da so sporočene hash vrednosti ostale konsistentne in končne ter c) dokumentirati ugotovljene anomalije. Naslednji pododdelki opišejo vsak posamezen test.

Testi formata grafičnih datotek

Testi formata grafičnih datotek so zahtevali, da se ciljno mobilno napravo napolni z grafičnimi datotekami (npr.: .jpg, .bmp, .gif) iz vnaprej določenega nabora podatkov z univerzalnim izmenjevalnikom Cellebrite UME-36. Cellebrit UME-36 je bil izbran izključno na podlagi razpoložljivosti in sheme prenosa podatkov. Enota Cellebrite UME-36 je samostojni prenos telefonskega pomnilnika in varnostno kopiranje.

MMS testi

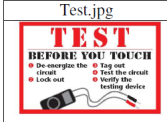
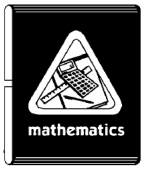
MMS testi so zahtevali mobilne naprave, ki so sposobne pošiljati in sprejemati sporočila MMS. MMS se uporablja za pošiljanje grafične datoteke na ciljne mobilne naprave. Ko je bilo sporočilo MMS uspešno sprejeto na ciljni mobilni napravi, je bila grafična datoteka shranjena v notranji pomnilnik ciljne mobilne naprave.

Bluetooth testi

Za teste Bluetooth so potrebne mobilne naprave, ki podpirajo Bluetooth. Forenzična delovna postaja je bila uporabljena za pošiljanje grafične datoteke z uporabo Bluetootha vsem ciljnim mobilnim napravam.

Testi kartice MicroSD

Za teste kartice MicroSD so bile uporabljene naslednje tehnike, ki temeljijo na zmogljivostih forenzičnih orodij za mobilne naprave in mobilnih naprav. Mobilne naprave, ki ne podpirajo MicroSD, so zahtevale shranjevanje grafične datoteke na flash pomnilnik in nato z Cellebrite UME-36 prenos v notranji pomnilnik mobilne naprave. Za mobilne naprave, ki podpirajo MicroSD in je bilo potrebno uporabiti Secure View, je bila grafična datoteka kopirana v notranji pomnilnik mobilne naprave iz kartice MicroSD. Pridobitev izve-

Test.jpg	Bluetooth.jpg	Card.jpg
		
3c3111ded5df821d66 8aecf9b598100b	6c8a1401a3af826450 4f16334e774b5c	77bebd7fb998797dd 5768c99fdbda8f6
Mathematics.bmp	Stress-test.gif	Mail.jpg
		
7d3b824769389bead b69b536a0295662	9b902382728b6bbdc 65009a5d1084041	d57fac85a5be5a7804 05a0484254256b

Slika 1: Vnaprej določen nabor podatkov (grafične datoteke).

dena z *Device Seizure*, je omogočila neposredno pridobitev grafične datoteke pomnilniške kartice MicroSD.

Testi ozadij

Testi ozadij so zahtevali, da ima ciljna mobilna naprava naloženo grafično datoteko .jpg iz vnaprej določenega nabora podatkov z uporabo Cellebrite UME-36. Ko je bila grafična datoteka uspešno shranjena v notranji pomnilnik mobilne naprave, je bila datoteka ročno dodeljena kot ozadje.

Testi kamere telefona

Testi kamere telefona so zahtevali mobilne naprave, ki vsebujejo notranjo kamero. Grafične datoteke, posnete z notranjim fotoaparatom, so bile prenesene na forenzično delovno postajo in mobilne naprave z uporabo Cellebrite UME-36.

5. REZULTATI

To poglavje povzema končne rezultate in nudi dodatne informacije o vsakem izvedenem testu. Zaradi omejitev v formatu grafičnih datotek na mobilnih napravah, rezultati testov za nekatere naprave morda ne vsebujejo hash vnosa za določen preskus.

Rezultati testov formata grafičnih datotek

Device Seizure in *Secure View* sta vračala dosledne hash vrednosti s forenzično delovno postajo.

Rezultati MMS testov

Ugotovljeno je bilo, da so hash vrednosti za poslane grafične datoteke neskladne v različnih družinah mobilnih naprav. Zaradi tega je prišlo do drugega kroga testov, da bi se preverilo, ali so bile ugotovitve hash neskladnosti povezane z različnimi implementacijami formata MMS za mobilne naprave.

Rezultati MMS testov - drugi krog

Drugi krog testov je potrdil, da: a) so bile neskladne hash vrednosti v obeh podobnih in različnih družinah mobilnih naprav in b) ponovno pošiljanje shranjenih grafičnih datotek z uporabo MMS, lahko povzroči različne hash vrednosti.

Rezultati Bluetooth testov

Device Seizure in *Secure View* sta vračala dosledne hash vrednosti s forenzično delovno postajo.

Rezultati testov kartice MicroSD

Device Seizure in *Secure View* sta vračala dosledne hash vrednosti s forenzično delovno postajo.

Rezultati testov ozadij

Forenzična orodja za mobilne naprave so ustvarile dosledne hash vrednosti za vse preizkušene mobilne naprave. Hash vrednosti, ki se jih je ustvarilo z forenzično delovno postajo, so se ujemale z vrednostmi, ki jih je vrnilo forenzično orodje za mobilne naprave.

Rezultati testov kamere telefona

Device Seizure in *Secure View* sta vračala dosledne hash vrednosti s forenzično delovno postajo.

6. ZAKLJUčEK

Cilj testov, opravljenih na Purdue University, je bil ugotoviti, ali hash vrednosti za grafične datoteke ostanejo skladne med forenzičnimi orodji za mobilne naprave in forenzično delovno postajo. Večina opravljenih testov (tj.: testi formata grafičnih datotek, Bluetooth testi, testi kartice MicroSD, testi ozadij, testi kamere telefona) so pokazali, da vrnjene vrednosti ostajajo konsistentne. Nedoslednosti pa se pojavijo, ko se grafične datoteke mobilne naprave prenašajo z MMS.

Z več kot 2 milijardami mobilnih telefonov, ki se uporabljajo danes, forenzika mobilnih naprav še naprej ostajajo glavno področje, ki je zanimivo za forenzično skupnost. Z razvojem mobilnih naprav se povečuje tudi zmogljivost shranjevanja in bogastvo podatkovnih objektov. Podatki, pridobljeni iz mobilnih naprav, so z raziskovalnega vidika pogosto koristni pri zagotavljanju sledi ali rešitvi primera. Zato je raziskovanje vedenja in zanesljivosti forenzičnih orodij mobilnih naprav ugodno za razvijalce in forenzično skupnost.

Medtem ko so bile opravljene minimalne raziskave o hash vrednostih, izračunanih za podatkovne objekte mobilnih naprav, bodo prihodnje raziskave raziskovale učinke dodatnih podatkovnih objektov (npr.: avdio, video, dokumenti), ki se najpogosteje nahajajo na mobilnih napravah, kar je bistvenega pomena.

7. REFERENCE

[1] Md5. `https://en.wikipedia.org/wiki/MD5`. Accessed: 2018-05-13.

[2] Sha-1. `https://en.wikipedia.org/wiki/SHA-1`. Accessed: 2018-05-13.

[3] R. Ayers. Guidelines on mobile device forensics (draft), 2013.

[4] P. Forensics. Device seizure v1.3>, 2007.

[5] R. V. D. Rizwan A. Study of cryptographic hashing: An analysis from the perspective of developing efficient generalized forensics framework for the mobile devices, 2012.

[6] N. Shachtman. Fighting crime with cellphones' clues., 2006.

[7] S. View. Secure view kit for forensics>. http://www.datapilot.com/productdetail/253/producthl/Notempty, 2009.

[8] H. Y. Xiaoyun W. How to break md5 and other hash functions, 2009.

Forenzična preiskava s pomočjo podatkov mobilnih operaterjev

Povzetek članka

Žiga Pintar
Univerza v Ljubljani, Fakulteta za računalništvo in informatiko

Nejc Rebernik
Univerza v Ljubljani, Fakulteta za računalništvo in informatiko

Povzetek

V tem poročilu bo govora o CDR (call detail record) oziroma o podrobnostih klica, ki se beležijo ob uporabi mobilnih telefonov v današnjem času. Pogledali si bomo, kaj ti podatki vsebujejo, kaj je njihov namen in tudi kako se pridobivajo in hranijo. Poročilo je nastalo na podlagi kazenskega primera [1], kjer so bili prav ti podatki uporabljeni na sodišču in so dokazali, da je oseba obtožena kaznivega dejanja krivo pričala glede svoje lokacije.

Ključne besede

Mobilna omrežja, CDR (call detail record), mobilni podatki in uporaba na sodišču, opis kazenskega postopka

1. UVOD

V današnjem svetu je postala uporaba mobilnih oziroma prenosnih telefonov skoraj nujna in neizbežna. Trenutno so postali del našega vsakdana in si s težavo predstavljamo, da bi bilo kako drugače. Vendar se veliko ljudi ne zaveda, kakšne podatke lahko hrani njihova mobilna naprava. Podatki so lahko shranjeni lokalno kot so tekstovna sporočila, zgodovina klicev, naložene aplikacije, zgodovina brskanja in tako dalje. Vendar to niso edini podatki, ki se lahko izkažejo za uporabne pri dokazovanju oziroma izpodbijanju krivde osumljene osebe. Pomemben vir so tudi zapisi povezav mobilnih naprav na antene prisotne na stolpu, med izvajanjem klica oziroma pošiljanjem sporočila. To so tako imenovane podrobnosti klica oziroma Call detail records - CDR in se hranijo pri operaterjih, ki imajo v lasti omrežje. Kot bo razvidno iz nadaljevanja poročila se bomo navezali na primer, ki so ga raziskovali v Ameriki in pri katerem je uporaba prav teh podatkov med sodnim procesom pokazala, da je obtoženi prikrival resnico glede svoje lokacije v času, ko se je zgodilo kaznivo dejanje.

V prvem delu se bomo osredotočili na delovanje stolpov ter njihovih anten, to vključuje njihovo zanesljivost, delovanje, testiranje ter zgradbo. Nato si bomo ogledali samo vsebino CDR podatkov in kateri od teh podatkov so lahko uporabni za sodišče pri dokazovanju krivde. Na koncu pa si bomo ogledali še področje mobilnih podatkov bolj v širšem, saj nas zanimajo še drugi primeri, kjer je uporaba mobilnih naprav bila uporabljena na sodišču.

V drugem delu pa si bomo podrobneje ogledali že prej omenjen primer iz Amerike, ko so kriminalisti uporabili podatke CDR na sodišču. Predstavljen bo zajem in nadaljnja analiza podatkov ter postopki, ki so jih kriminalisti ubrali za dokazovanje njihove kredibilnosti na sodišču.

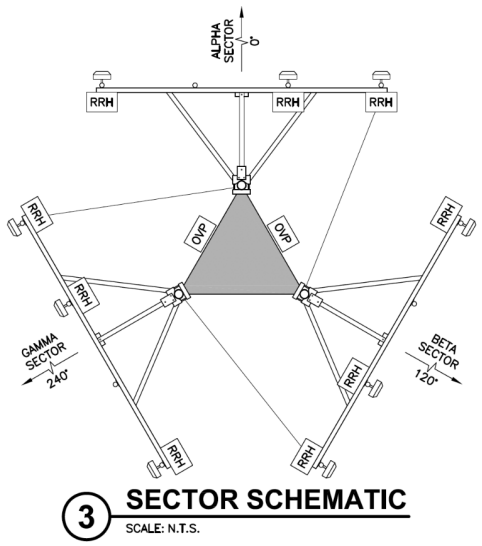
1.1 Analiza stolpov in anten

Mobilne naprave potrebujejo za klicanje ali pošiljanje sporočil predhodno vzpostavitev povezave z eno od anten, ki so prisotne na bljižnjem stolpu. Pokritost oziroma doseg posameznega stolpa se razlikuje, saj na njega vpliva več različnih naravnih kot tudi tehničnih pogojev. Naravni pogoji, ki lahko nekoliko vplivajo na delovanje stolpov in posledično na njihovo zmožnost povezovanja so:

- Trenutne vremenske razmere
- Višina stolpa gleda na teren v okolici
- Prisotnost listov na drevesih v okolici

Prav tako imajo lahko posamezni stolpi različno moč glede na to, kje se nahajajo. Lahko se nahajaljo na bolj ruralnih področjih, kjer so stolpi postavljeni bolj na redko, vendar so ti stolpi ponavadi močnejši in imajo zato večji doseg. V urbanih naseljih je povprečna pokritost stolpa približno 3 kilometre, medtem ko imajo stolpi na bolj neposeljenih lokacijah lahko pokritost tudi do 10 kilometrov.

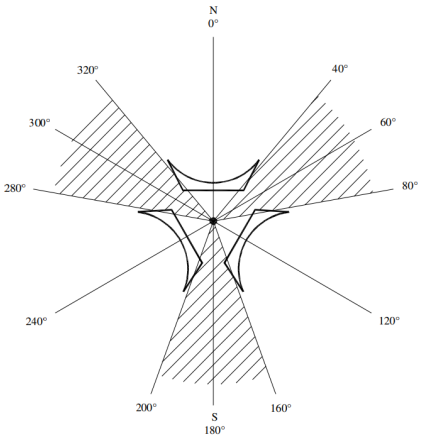
Tipičen stolp [1] [2] je sestavljen iz treh glavnih anten, ki so vse obrnjene v različne smeri. Vsaka od anten pokriva približno 120 stopinj kroga okoli stolpa. Prva in glavna antena, imenovana tudi alfa antena, je ponavadi usmerjena proti severu in predstavlja 0 stopinj. Njena pokritost se razteza od 300 stopinj oziroma -60 stopinj na levo in 60 stopinj na desno. Druga antena imenovana beta antena, je postavljena na 120 stopinj in je s tem usmerjana proti jugovzhodu. Pokritost beta antene se razteza od 60 do 180 stopinj. Zadnja izmed anten je gama antena, ki leži na kotu 240 stopinj in tako kaže proti jugozahodu. Na sliki 1 so prej omenjene antene in njihove usmeritve prikazane še grafično.



Slika 1: Na sliki vidimo posamezne sektorje (antene) in njihove smeri.

Ta postavitev anten ni absolutna, saj je lahko celoten stolp zasukan za določen kot. Razlogi za zamik stolpa so lahko, da je glede na geografsko okolico bolj primerna takšna postavitev ali pa je zaradi lege urbanih naselji boljša pokritost na podanem območju.

Ker tu govorimo o signalih se seveda pojavljajo območja, kjer se prekrivajo posamezne antene. To območje naj bi bilo veliko približno 40 stopinj, kar pomeni da se alfa antena nahaja za 20 stopinj v območju beta in se istočasno tudi beta nahaja za 20 stopinj v območju alfa. Prav ta pojav je bil ključnega pomena, da so morala biti izvedana testiranja prekrivanja območji in kakšne vpliv imajo ta na podatke. Pojav je prikazan na sliki 2, ki je vzeta iz poročila [1] na katerega se navezujemo.



Slika 2: Območja prekrivanja posameznih anten prikazana grafično.

1.2 CDR - podrobnosti o klicu

Ko se vzpostavi povezava med mobilno napravo in stolpom, oziroma eno od njegovih anten, se zabeležijo podatki [3], do katerih lahko dostopajo načeloma le operaterji. Podatki so lahko dostopni tudi sodišču vendar le v primeru, da ima dovolj velik razlog za pridobitev sodnega naloga. Ti podatki so lahko pomembni za sodne preiskave saj nosijo veliko informacij, ki lahko služijo kot sredstvo za dokazovanje oziroma izpodbijanje preostalih dokaznih gradiv v preiskavi.

CDR podatki med drugim vsebujejo pomembne informacije kot so:

- Telefonska številka s katere je bila opravljena storitev (klic, sporočilo,..)
- Telefonska številka, ki je bila prejemnik te storitve (klic, sporočilo,..)
- Kdaj se je klic začel (datum in čas)
- Trajanje klica
- Kje se je klic povezal in kje prekinil (ob morebitnem premikanju osebe med klicem)
- Tip storitve, ki je bil opravljena (klic, spročilo,..)
- Kateri stolp in katera antena na njem je bila uporabljena

Kakor je razvidno iz zgoraj navedenih podatkov je količina uporabnih informacij, ki jih lahko pridobimo iz CDR velika. Kot bomo videli v nadaljevanju, bodo podatki o anteni na katero je bil klic vezan ključnega pomena za sodno preiskavo.

1.2.1 Problemi CDR podatkov

Problem pri interpretaciji teh podatkov je velikost območja izvora, saj je ta ponavadi precej velik. Če kot primer vzamemo 160 stopinj (še 20 stopinj prekrivanja v vsako od sosednjih polji) in da ima stolp dosega približno 3 kilometre, to predstavlja kar 12 kvadratnih kilometrov od koder je klic lahko bil izveden. Vendar je tudi tako veliko območje še vedno uporabno ob primerni interpretaciji in predstavitvi.

Prav tako se pojavalja problem tudi zaradi gostovanja večih mobilnih operaterjev na enem stolpu. To pomeni, da so lahko podatki shranjeni v drugačnem formatu in za različna časovna obdobja. To pa lahko zelo oteži preiskavo kriminalistov, saj je zaradi tega potrebno postopati pri vsakem operaterju drugače.

Zaradi zaprtega sveta mobilnih komunikaciji je dostop do teh podatkov otežen in je zato težko podati dober primer oziroma prikaz resničnih CDR podatkov. Na spletu obstajajo različni urejevalniki, a tem v veliko primerih manjkajo določeni podatki in jih zato v tem poročilu ne bomo navajali.

1.3 Pregled področja

Na področju mobilnih naprav se je prav zaradi povečanega števila uporabnikov začelo pojavljati vedno večje število kaznivih dejanj, kjer je bila naprava na tak ali drugačen način uporabljena za kaznivo dejanje. Trenutno največji problem

s katerim se srečujejo kriminalisti je, da pri telefonih še ne obstajajo definirani standardi na mnogih področjih, kar otežuje zbiranje in analiziranje dokazov [2].

Kot zanimiv primer uporabe CDR podatkov so prometne nesreče. Sodišče lahko pridobi nalog za vpogled v mobilne storitve opravljene s strani storilca nesreče z namenom, da preverijo ali je ta med vožnjo telefoniral. To je zelo hitro razvidno, saj je potrebno preveriti le časovne podatke klica in nesreče. Če se čas ujema lahko z veliko vrjetnostjo rečemo, da je storilec med nesrečo opravljal klic in je zato kazensko odgovoren za nastalo nesrečo.

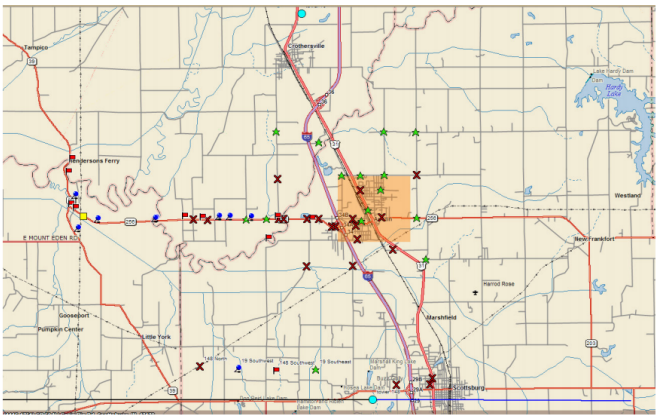
CDR podatki so lahko uporabni tudi pri odkrivanju goljufij [4]. V realnem času z preverjanjem oziroma iskanjem določenih sekvenc oziroma zaporedji lahko preventivno preprečijo morebitne goljufije. Prav tako pa se lahko uporabijo podatki po samem dejanju, kot so goljufije ko morajo osebe poslati sporočilo oziroma poklicati določeno številko, kjer se zaračunajo večji zneski za opravljeno storitev. V teh primerih lahko preiskovalci vidijo kam se klic veže, katero območje itd.

2. POVZETEK KAZNSKEGA POSTOPKA

V povzetem članku je sodišče preverjalo veljavnost alibija obtoženca, ta je trdil, da se v času kaznivega dejanja ni nahajal na kraju zločina. Kot smo že povedali, je lahko področje, ki ga lahko pokriva ena antena 12 kvadratnih kilometrov oziroma še več, če v okolici ni veliko zgradb, hribov ali dreves. V konkretnem kazenskem postopku, je šlo za urbano okolje, kjer je po izračunih območje pokrito z eno od anten na stolpu okrog 12 kvadratnih kilometrov. S takšnimi pogoji in podatki ni mogoče točno določiti lokacije iz katere je bil opravljen klic. Območje, ki smo ga izračunali je mogoče uporabiti le za zavračanje oziroma potrjevanje izjave obtoženca. Če obtoženi trdi, da se je ob določenem času nahajal na specifičnem mestu in je takrat opravil klic, sta možna dva scenarija. Obtoženi lahko trdi, da je klic opravil iz lokacije, ki se nahaja znotraj izračunanega območja in zato v tem primeru ni mogoče zanesljivo potrditi izjave, prav tako pa je ni mogoče zavreči. V drugem scenariju lahko obtoženi trdi, da je klic opravil iz lokacije, ki se nahaja izven izračunanega območja dosega antene. V takšnem primeru lahko sodišče na podlagi strokovnega mnenja izvedenca ovrže izjavo in alibi, ker se navedena lokacija klica ne sklada z izračunanim zbranimi dokazi.

V primeru iz članka je ekipa s strokovnim izvedencem organizirala teste za izračun dosega antene v kar se da podobnih pogojih, kot so bili na dan dogodka. Pridobili so enak mobilni telefon in naročniški paket pri istem mobilnem operaterju. Teste so opravljali ob podobnem času, da bi čimbolj poustvarili zasedenost omrežja in druge faktorje, ki bi lahko vplivali na rezultate. Zasedenost je pomembna saj se ob preobremenjenosti antene, ki je vzrok prevelikega števila odjemalcev, lahko klic preusmeri na drugo anteno na istem stolpu oziroma v skrajnem primeru kar na popolnoma drug stolp. Preiskovalci so poustvarili čim več faktorjev, ki so bili prisotni na dan zločina, vendar so veter, drevesa in zasedenost omrežja bili lahko le delno poustvarjeni. Pogozditev se lahko tekom preiskave, sploh če ta traja več let spremeni, na kar preiskovalci nimajo vpliva. Zasedenost omrežja lahko ponovimo samo približno, saj se število naročnikov lahko poveča ali zmanjša. Najboljši približek je opravljanje klica ob isti uri, tako namreč dosežemo približek količini prometa, ki se dnevno giblje po dovolj predvidljivih vzorcih.

V konkretni preiskavi je bilo opravljenih 65 klicev, na 65 različnih lokacijah. CDR podatke so pridobili od mobilnega operaterja z sodnim nalogom okrajnega sodišča. Ko so pridobili podatke so na zemljevid mapirali lokacije petih anten in 65 opravljenih klicev. Slika 3 prikazuje območje na katerem so bili izvedeni klici.



Slika 3: Na sliki vidimo področje raziskovanja in označena mesta zanimanja za kriminaliste.

Stolp 148 s tremi in stolp 19 z dvemi antenami na katerih so bili registrirani klici, imajo ločene simbole glede na to na kateri anteni je bil klic sprejet. Več kot polovico testnih klicev je bilo opravljenih na cesti na kateri je osumljenec trdil, da se je v času zločina nahajal. Preostali testni klici so bili opravljeni na naključno izbranih lokacijah kot so križišča, parki in druga območja z visoko uporabo mobilnih telefonov.

Sodišče je sodelovalo pri preiskavi ekipe s strokovnim izvedencem, tako da je od mobilnega operaterja pridobilo CDR podatke o 65 testnih klicih, ki jih je izvedla preiskovalna ekipa in o klicih opravljenih s strani osumljenca. V Ameriki so vsi mobilni operaterji zavezani k sodelovanju s policijo in preiskovalnimi organi vendar le ob predložitvi primernega razloga za vdor v posameznikovo zasebnost. Za razliko od navadnih uporabnikov in naročnikov, ki lahko pridobijo samo čas in prve tri številke prejetih in klicanih številk lahko sodišče z sodnim nalogom pridobi podrobne tehnične podatke o številki, točni anteni in drugih podrobnostih, kot so to storili tudi v preiskavi iz poročila.

Sodišče se je po posvetovanju s strokovnim izvedencem lotilo nadaljevanja sodnega postopka. Zavedali so se omejitev izsledkov pridobljenih s testnimi klici, a so tudi s temi podatki pridobili močnejšo sodno podlago za obravnavo. Obtoženi je trdil, da se je na večer umora nahajal na 6 različnih lokacijah. Trditev je bila, da je opravil klic v jugozahodnem delu mesta vendar so podatki strokovne preiskave pokazali drugače. Zaradi opisanih tehničnih razlogov, je sodišče uspešno izpodbilo osumljenčevo izjavo. Alibi osumljenca se ni obdržal, saj je preiskava pokazala, da klic zaradi postavitve anten ni mogel izvirati iz jugozahodnega dela mesta. Sodi-

šče je uspelo dokazati krivdo za umor. V tem primeru so bili podatki strokovne preiskave uspešni na sodišču, ker se jih je dalo predstaviti tako, da so ovrgli osumljenčev alibi. V primeru potrebe po natančni lokaciji izvedenga klica, takšen način sledenja ne bi bil uspešen. Z takšno strokovno analizo lahko pridobimo uporabne podatke za sodni pregon, uspešnost in uporaba pa je odvisna od konteksta uporabe in njegove predstavitve.

3. ZAKLJUčEK

Analiza postavitve mobilnih telefonskih anten lahko znatno pomaga pri pridobivanju podatkov za sodni pregon. Uporaba teh dokazov je odvisna od konteksta in načina preiskave. Ne omogoča točnega določanja lokacije, omogoča pa zavrnitev lokacij, ki zaradi postavitve anten niso možne. Uporaba teh ugotovitev v sodiščih je odvisna od primera do primera in ni vedno prava izbira. V primerih, kjer se lahko uporabi pa je v veliko pomoč sodišču.

4. REFERENCES

[1] Terrence P. lO'Connor. Provider side cell phone forensics. *Small scale digital device forensics journal*, 3(1), June 2009.

[2] The officer: The other side of mobile forensics. Dosegljivo: `https://www.officer.com/home/article/10248785/the-other-side-of-mobile-forensics`. [Dostopano: 11. 5. 2018].

[3] Call detail record. `https://en.wikipedia.org/wiki/Call_detail_record`, 2018.

[4] How is cdr data used? `https://www.quora.com/How-is-CDR-data-used`, 2016.

FORENZIKA ANDROIDA

POENOSTAVITEV PREGLEDOVANJA MOBILNIH NAPRAV

Aljaž Blažej
Fakulteta za računalništvo in informatiko
63140018
ab4468@student.uni-lj.si

Žan Ožbot
Fakulteta za računalništvo in informatiko
63140183
zo4120@student.uni-lj.si

POVZETEK

Pametne mobilne naprave so postale pravi superračunalniki. Te nam pomagajo pri vsakodnevnih opravilih ter pri tem beležijo ogromne količine podatkov, ki lahko postanejo velik vzvod v primeru, da pristanejo v napačnih rokah. Cilj članka je bralcu pokazati kako enostavno najdemo nekatere pomembne podatke in ga s tem prepričati kako pomembno je imeti ustrezno zavarovano mobilno napravo. Osredotočimo se na naprave, ki poganjajo operacijski sistem Android. Slednjega na kratko tudi opišemo. Srednji del članka je posvečen povzetku **Android Forensics: Simplifying Cell Phone Examinations** ter predstavitvi njihovih metod preiskovanja in analize. Na koncu opišemo tudi metode in rezultate našega poskusa preiskovanja in analize naprave Nexus 4. Najdeni podatki so nas nedvomno presenetili.

KLJUČNE BESEDE

Digitalna forenzika, Android, mobilne naprave

1. UVOD

Število pametnih mobilnih naprav se iz dneva v dan povečuje. Naprave postajajo vedno bolj zmogljive ter vsak dan obdelajo na milijone podatkov ter beležijo vsako človeško interakcijo. Prav zaradi slednje lastnosti so pri forenzičnih preiskavah celo bolj priljubljene od osebnih računalnikov. Večina mobilnih naprav vsebuje več informacij glede na preiskan bajt, kot katera koli druga naprava [6]. Že podatek, kot je sporočilo SMS/MMS, zgodovina klicev ali zgodovina obiskanih spletnih strani, lahko spremeni potek same preiskave. Američani so vedno bolj povezani v svet digitalnih informacij. Po zadnjih raziskavah si jih kar 77% lasti pameten mobilni telefon [4]. Njihove naprave večinoma poganja operacijski sistem Android, katerega tržni delež je po pravilnih napovedih avtorjev članka **Android Forensics: Simplifying Cell Phone Examinations** narasel na skoraj 76% in tako prehitel iOS ter Blackberry [7, 6]. V delu smo povzeli metode prej omenjenega članka, ki se nanašajo na pridobivanje in analizo podatkov na mobilnih napravah Android. Pri tem smo se poslužili številnih orodij, ki so nam delo precej poenostavila. Na koncu smo celotno napravo analizirali tudi fizično ter odkrili, da Android le ni tako varen kot si to misli navaden uporabnik.

2. SPLOŠNO O ANDROIDU

Namen članka je bralcu kar se da razumljivo predstaviti nadaljnja pogljavja, ki obsegajo pridobitev in analizo podatkov. Da bi uresničili podan namen, smo sprva na kratko preleteli zgodovino, arhitekturo in v nadaljevanju predstavili nekatere lastnosti operacijskega sistema Android.

2.1 ZGODOVINA

Android je bil sprva mišljen kot operacijski sistem, ki bi tekel na digitalnih kamerah, vendar so razvijalci kaj kmalu ugotovili, da targetirajo premajhen trg. S tem so se takrat postavili na rob velikanom kot sta Symbian in Microsoft Windows Mobile [9]. Jedro operacijskega sistema je odprtokodno in je bilo prvotno razvito znotraj podjetja Android Inc., ki ga je v letu 2005 prevzelo podjetje Google.

Prva produkcijska različica Androida je bila naložena na telefon HTC Dream. Od leta 2008 naprej pa imajo v Googlu navado vsako novo različico poimenovati po sladici, saj menijo, da nam te naprave delajo življenje slajše [9]. Tako so najstarejše verzije poimenovane Cupcake, Donut, Eclair in Froyo. Leta 2007 so se velikani kot so Google, HTC, Motorola, Samsung in drugi združili v konzorcij Open Handset Alliance z namenom razviti prvo celovito rešitev za mobilne naprave [9].

2.2 ARHITEKTURA

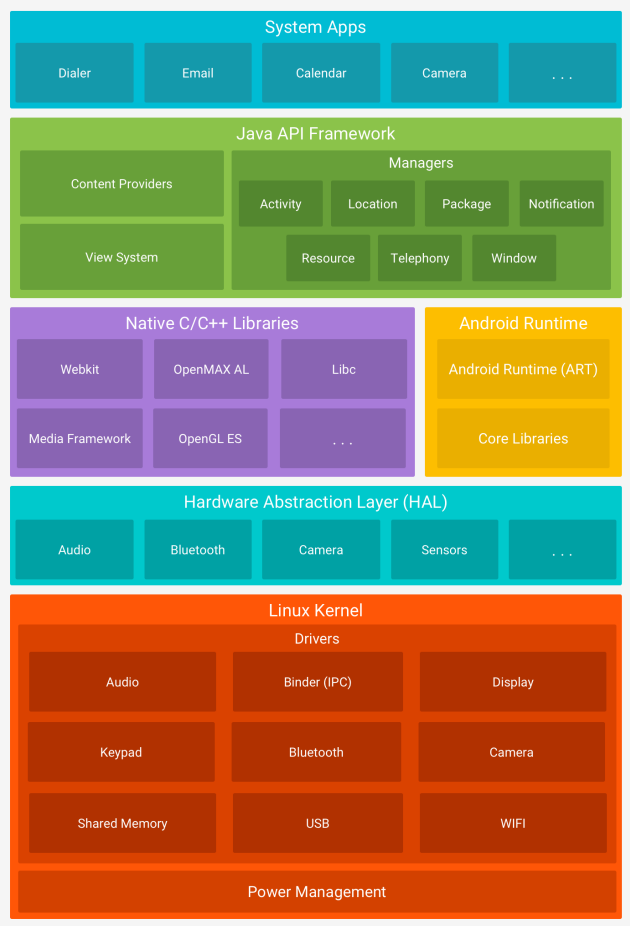
Android je odprtokodna programska oprema, ki temelji na Linuxu, narejena z namenom podpreti široko paleto naprav [5]. Slika 1 prikazuje vse večje komponente znotraj platforme Android. Kot je razvidno iz slike, Android sestavljajo jedro Linux, abstrakcijski sloj strojne opreme, Android Runtime, izvorne C/C++ knjižnice, okvir Java API ter nabor sistemskih aplikacij.

Uporaba jedra Linux omogoča operacijskemu sistemu Android izkoriščanje varnostne funkcionalnosti in omogočanje razvijalcem strojne opreme lažjo izdelavo gonilnikov [5]. Abstrakcijski sloj strojne opreme ponuja standardni vmesnik, ki izpostavi strojne zmogljivosti visokonivojskemu okvirju Java API.

Predhodnik Android Runtime je Dalvik (pred Andorid ver-

zijo 5.0) [5], ki je že takrat omogočal več aplikacijam hkratno delovanje, tako, da je vsaka aplikacija tekla v svojem lastnem ločenem virtualnem stroju [6].

Mnoge sistemske komponente so napisane v izvorni kodi C/C++. Nekatere od teh komponent so tudi izpostavljene preko okvirja Java API.



Slika 1: Arhitektura operacijskega sistema Android.

2.3 LASTNOSTI

Eden izmed največjih virov podatkov so podatkovne baze SQLite [6], ki jih dandanes za shranjevanje podatkov uporablja velika večina aplikacij. Najdemo jih v lahko ali v notranjem pomnilniku naprave ali na kartici microSD.

Podprte različice datotečnih sistemov se lahko razlikujejo z vsako napravo Android. Najpogostejši pa so naslednji (i) exFAT (ii) F2FS (iii) JFFS2 in (iv) YAFFS2. Poleg pomnilniških (angl. flash) datotečnih sistemov, Android podpira tudi (i) EXT2/EXT3/EXT4 (ii) MSDOS ter (iii) VFAT, ki jih ponavadi najdemo na medijih, kot npr. kartica microSD [1].

Varnostni mehanizmi aplikacij sistema Android temeljijo na dovoljenjih (angl. permissions). Zaradi narave virtualnih strojev, znotraj katerih aplikacije tečejo, aplikacije ne morejo dostopati do podatkov drugih aplikacij, razen v primeru, da eksplicitno dobijo dovoljenje za to početje [6].

3. 'ROOTANJE' NAPRAVE IN IZDELAVA SLIKE

Metode pridobivanja dostopa do korenskega imenika ("rootanja") se med napravami in verzijami Androida razlikujejo [8]. Izvorni članek [6] opisuje primer rootanja naprave Sprint HTC Hero na verziji Androida 1.5 s pomočjo orodja **AsRoot2**. Avtorji se zgledujejo po postopku, opisanem na forumu XDA Developers (`https://forum.xda-developers.com/`), kjer lahko najdemo tudi opise za druge naprave in verzije.

Napravi moramo sprva zamenjati SD kartico z novo, saj proces spreminja vsebino kartice ter s tem uniči morebitne dokaze. Naslednji korak je namestitev orodja Android SDK (*Android Software Development Kit*) na računalnik, s katerim se želimo povezati z napravo. Orodje vsebuje vse potrebne gonilnike, ki jih potrebujemo za povezavo mobilne naprave z računalnikom (preko mostička ADB). Po priklopu naprave se v lupini pomaknemo v mapo `AndroidSDK/tools` in poženemo sledeči ukaz:

```
$ adb devices
```

Ukaz nam izpiše priklopljene naprave in njihove serijske številke.

Zatem lahko pričnemo s procesom rootanja. Prenesemo arhiv programa AsRoot2 in v lupini poženemo naslednje ukaze:

```
$ adb push asroot2 /data/local/
$ adb shell chmod 0755 /data/local/asroot2
$ adb shell
$ /data/local/asroot2 /system/bin/sh
$ mount -o remount,rw -t yaffs2
↪  /dev/block/mtdblock3 /system
$ cd /system/bin
$ cat sh > su
$ chmod 4755 su
```

Če nismo naleteli na nobeno napako, bi morali imeti pravice za dostop do korenskih direktorijev in s tem možnost izdelave slike podatkov.

3.1 IZDELAVA SLIKE PODATKOV

Datotečni sistem na napravah Android je porazdeljen po mapi `/dev`. Posamezne datoteke se med napravami razlikujejo. Poimenovanje in opis datotek na opisani napravi je sledeč:

- `mtd0` je zadolžena za razna opravila
- `mtd1` vsebuje obnovitveno sliko
- `mtd2` vsebuje zagonsko particijo
- `mtd3` vsebuje sistemske datoteke

- `mtd4` vsebuje predpomnilnik
- `mtd5` vsebuje uporabniške podatke

Kljub temu, da lahko katerakoli od zgoraj omenjenih datotek vsebuje podatke, pomembne za forenzično raziskavo, se bomo v nadaljevanju posvetili podatkom iz `mtd3` in `mtd5`.

Za kopiranje slike podatkov na SD kartico, moramo sprva odpreti lupino adb in nato pognati naslednji ukaz:

```
dd if=/dev/mtd/mtd0 of=/sdcard/mtd0.dd bs=1024
```

Ukaz poženemo za vseh šest datotek in s tem na SD kartici ustvarimo slike, ki zajemajo celoten pomnilnik. Pri tem je priporočljiva uporaba blokatorja pisanja (angl. write-blocker).

4. PREISKAVA SLIK POMNILNIKA

Pri pregledu datotek so si avtorji članka pomagali z orodjem Access Data's Forensic Tool Kit (FTK) v1.81. Za to orodje so se odločili, ker omogoča dobro obrezovanje (angl. carving) in iskanje po datotekah.

Orodje FTK je bilo nastavljeno tako, da je omogočalo polno indeksiranje in obrezovanje podatkov nad vsemi šestimi slikami. Ker je bila naprava predhodno vsakodnevno uporabljena kakšna dva meseca, je bilo rezultatov kar veliko. Najdenih je bilo 207 dokumentov tipa PDF in HTML in 12.709 slikovnih datotek tipa BMP, GIF, JPEG in PNG.

4.1 NAJDENI DOKUMENTI

Večina najdenih dokumentov je bila za forenzično raziskavo nezanimivih, saj so HTML dokumenti v večini vsebovali reklamni material. Najden pa je bil koristen dokument tipa PDF, ki je bil žal zelo fragmentiran. Kljub temu, da dokumenta ni bilo mogoče odpreti s programom Acrobat Reader, je FTK prikazal del njegove vsebine. Dokument je vseboval sporočila, imenik, zgodovino brskanja, podatke iz Facebooka, obiskane YouTube posnetke in glasbo.

4.2 NAJDENO SLIKOVNO GRADIVO

Prav tako, kot pri dokumentih, je tudi večina slikovnih datotek forenzično nepomembnih. Slika `mtd3.dd` je vsebovala slikovne datoteke aplikacij, kot so igre, vreme, tipkovnica, in veliko število ikon. Slika `mtd4.dd` pa je v večini vsebovala predpomnjene datoteke. Najdenih je bilo 30 slik iz uporabnikovega računa Gmail. Bile so zelo fragmentirane in zato le nekatere uporabne. Večina zanimivih datotek je bilo najdena na sliki `mtd5.dd`, ki je vsebovala uporabniške podatke. Najdenih je bilo kar nekaj slik, zajetih z vgrajenim fotoaparatom ter slik prenesenih iz spleta in aplikacij, kot so Facebook, YouTube, Pandora ter SprintTV. Prav tako je bilo najdenih nekaj slik iz sporočil MMS in ikon.

4.3 ROČNO ISKANJE

Ker je orodje FTK našlo zelo veliko količino podatkov, so jih avtorji članka filtrirali z iskalnimi nizi. Primer takega niza je uporabnikov e-poštni naslov, s katerim so prišli do 1628 zadetkov. Zadetki, pridobljeni iz spletnega e-poštnega odjemalca, so vsebovali veliko tujih e-poštnih naslovov in vsebine pošte. S pomočjo drugih iskalnih nizov je bilo najdenih tudi nekaj URL naslovov spletnih strani in gesel, ki jih je uporabnik na teh straneh uporabljal. Gesla niso bila šifrirana, kar je za forenzično preiskavo zelo koristno.

5. PREISKAVA Z ORODJI ADB IN CELLEBRITE

S pomočjo orodja FTK je bilo pridobljenih veliko koristnih podatkov, vendar je bila večina močno fragmentiranih in neberljivih. Zato je pomemben tudi pregled datotečnega sistema naprave s pomočjo orodja `adb`.

Večino raziskave je bilo izvedene v mapi `/data/data`, ki je vsebovala 154 podmap s podatkovnimi bazami. Posamezne podatkovne baze so bile prenešene na SD kartico z ukazom:

```
dd if=/data/data/subdir/databases/file.db
↪  of=/sdcard/file.db
```

Veliko od pregledanih podatkovnih baz je vsebovalo zanimive podatke. Baza aplikacije *Peep* (HTCjeva aplikacija za Twitter), ki se je nahajala na naslovu `/data/data/com.htc.htctwitter/databases/htcchrip.db`, je vsebovala podatke o uporabniškemu računu, šifrirano geslo in seznam oseb, ki jim uporabnik sledi. Najdenih je bilo tudi 1460 sporočil, ki je vsebovalo podatke o pošiljatelju in polje, ki je podalo informacijo, če je sporočilo zasebno ali javen "tvit".

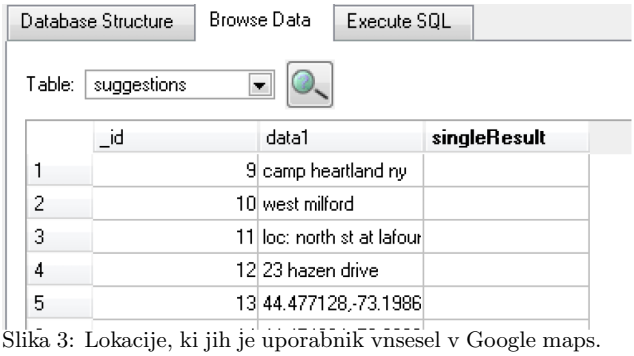
Naslednja zanimiva podatkovna baza se je nahajala na naslovu `/data/data/com.android.browser/databases/browser.db` in vsebovala podatke Androidovega brskalnika. Podatki so bili sestavljeni iz uporabniških imen, URL naslovov obiskanih strani, nešifriranih gesel in zgodovine brskanja (slika 2).

237	237	where the wild partie	1259461432984
238	238	pulp fiction soundtra	1259465046279
239	239	ball and chain lyrics	1259469379237
240	240	sublime scarlet bego	1259469890406
241	241	party hard lyrics	1259472218623

Slika 2: Zgodovina brskanja.

Pridobljena je bila tudi zadnja zabeležena lokacija in sicer v datoteki `/data/data/com.android.browser/gears/geolocation.db`.

Iz podatkovne baze Google maps (`/data/data/com.google.android.apps.maps/databases/search_history.db` so bili pridobljeni nizi iskanja lokacij, ki jih je vnesel uporabnik (slika 3).



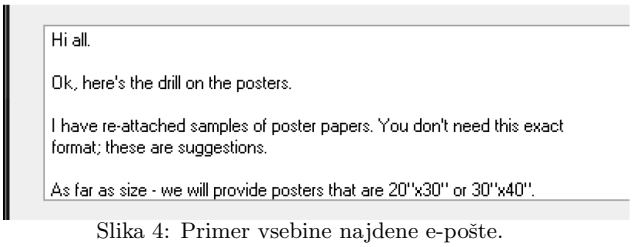
Slika 3: Lokacije, ki jih je uporabnik vnesel v Google maps.

Zelo pomembna podatkovna baza je bile najdena v mapi `/data/data/com.android.providers.telephony/databases/`. Baza `mmssms.db` je vsebovala sporočila SMS in MMS ter njihove pošiljatelje ter prejemnike.

Najdenih je bilo tudi nekaj zbrisanih sporočil ter zvočnih sporočil tipa *AMR* (Adaptive Multi-Rate), ki so se nahajali v mapi `/data/data/com.coremobility.app.vnotes/files`.

V mapi Sprintove navigacijske aplikacije Telenav (`/data/data/com.telenav.app.android.sprint/files`) se je nahajalo kar nekaj datotek, povezanih z lokacijskimi podatki, od katerih je bila najbolj koristna `ANDROID_TN55_recent_stops.dat`. Vsebovala je podatke zadnjih lokacij, ki jih je obiskal uporabnik.

Podatki o e-pošti so bili najdeni v mapi aplikacije Gmail (`/data/data/com.google.android.providers.gmail/databases`). V mapi je bila najdena podatkovna baza, ki je vsebovala podatke o pošiljatelju in prejemniku ter telo (slika 4) in zadevo pošte.



Slika 4: Primer vsebine najdene e-pošte.

Zadnja od najdenih podatkovnih baz je baza `contacts.db`, ki se je nahajala v mapi `/data/data/com.android.providers.contacts/databases/`. Vsebovala je zgodovino klicev, ki je bila sestavljena iz telefonskih številk, datumov in dolžine klica. Prav tako so bile v bazi najdene informacije o kontaktih in število klicev, ki jih je uporabnik z njimi opravil.

5.1 ANALIZA Z ORODJEM CELLEBRITE

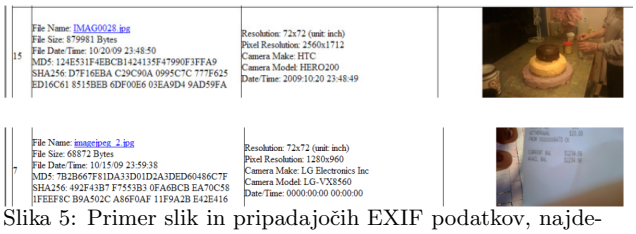
Pri analizi telefona je bila uporabljena tudi forenzična naprava CelleBrite Universal Forensic Extraction Device (UFED). Naprava UFED je namenjena avtomatskemu pridobivanju podatkov iz mobilnih telefonov, kot so naslovi, slike, kontakti, glasba, videi, sporočila, zgodovina klicev in indentifikacijski podatki. S telefonom komunicira preko omogočenje podatkovnega kabla, infrardeče povezave (IR) ali Bluetooth povezave (BT). Podatke iz kartice SIM lahko pridobi preko povezave ali s fizičnim priklopom le-te. UFED ima prav tako vgrajen blokator pisanja, s katerim prepreči morebitno uničenje podatkov.

Če želimo telefon HTC Hero povezati z UFED, moramo na telefonu sprva vklopiti USB razhroščevanje (angl. USB debugging). UFED vodi forenzika skozi korake, ki so potrebni za zajem podatkov. Končne rezultate nato naprava izvozi na USB ključ v obliki dokumenta HTML.

Poročilo, ki ga izdela naprava UFED, se začne z osnovnimi informacijami mobilne naprave, kot so ime modela, programske opreme, identifikator MEID in datum zbiranja podatkov. Temu pa sledijo drugi podatki, kot so:

- sporočila SMS (najdenih 1070)
- kontakti (najdenih 56)
- vhodni klici (najdenih 107)
- izhodni klici (najdenih 192)
- zgrešeni klici (najdenih 49)
- fotografije (najdenih 69)
- video posnetki (najden 1)

Naprava je našla vse izmed podatkov, ki so jih avtorji članka našli ročno.



Slika 5: Primer slik in pripadajočih EXIF podatkov, najdenih z orodjem UFED.

6. POVZETEK REZULTATOV

Prikazani pristopi ne zajemajo vseh potrebnih postopkov, ki jih je potrebno izvesti pri forenzični raziskavi mobilne naprave, je pa z njimi mogoče pridobiti kar nekaj zanimivih podatkov. V naslednjem razdelku so napisane prednosti in slabosti posameznega pristopa:

- Analiza z orodjem FTK
  - **Prednosti**: Najdena izbrisana sporočila in kontakti, ki jih druge metode niso pridobile. Najdena gesla.
  - **Slabosti**: Zahteva dostop do korenskega imenika, rezultati so bili močno fragmentirani.
- Preiskava z orodjem adb

  - **Prednosti**: Najdeni praktično vsi podatki, ki so pomembni za forenzično raziskavo, kot so zgodovina klicev in brskanja, fotografije, sporočila, e-pošta, podatki GPS, zvočna sporočila ter gesla.
  - **Slabosti**: Zahteva dostop do korenskega imenika in ne najde vseh izbrisanih sporočil, kontakov ter zgodovine klicov.
- Analiza z orodjem CelleBrite
  - **Prednosti**: Zelo preprosta metoda, ki najde sporočila, zgodovino klicov, fotografije, videe in kontakte.
  - **Slabosti**: Ne najde e-pošte in zgodovine brskanja.

Izkazalo se je, da je izmed metod najbolj koristna preiskava z orodjem adb, s katero iz številnih podatkovnih baz lahko pridobimo pomembne informacije. Pri raziskavi je seveda ključen tudi zajem slike pomnilnika, saj brez tega ne moramo najti izbrisanih datotek.

7. PREISKAVA NOVEJŠE NAPRAVE

Eden izmed vodilnih pametnih telefonov, Sprint HTC Hero, ki sta ga avtorja uporabila v njuni preiskavi, se danes smatra kot zastarela naprava. Kljub svoji starosti, še vedno vsebuje širok nabor podatkov, ki jih lahko pridobimo pri forenzični preiskavi. Tudi v tem članku smo se poskusili čim bolje držati postopka opisanega v zgornjih poglavjih. Uporabili smo Nexus 4, na katerem je naložen operacijski sistem CyanogenMod. Slednji temelji na operacijskem sistemu Android, verzija 6.0 - Marshmallow. Pametni telefon je kot nalašč za forenzično preiskavo, saj ima poškodovan zaslon do te mere, da ga ni moč uporabljati. Naprava je že imela vse potrebno za začetek izvajanja preiskovanja podatkov. 'Rootatajne' in omogočenje USB razhroščevanja ni bilo potrebno, saj je to delo opravil uporabnik sam v postopku nalaganja operacijskega sistema CyanogenMod.

Slika 6: Simbolična slika naprave Nexus 4.

7.1 IZDELAVA SLIKE PODATKOV

Naprava ne vsebuje reže za spominsko kartico microSD, zato smo morali slike točk interesa sprva shraniti na notranji pomnilnik in nato z ukazom `adb pull` prenesti na računalnik.

S tem početjem smo lahko pokvarili kakšen izbrisan podatek, vendar je bil to edini način, da smo lahko pridobili slike točk interesa. Preiskali smo mape, vidne na seznamu 1, za katere menimo, da vsebujejo največ podatkov, ki bi lahko bili pomembni pri forenzični preiskavi.

- `/cache` vsebuje predpomnilnik
- `/system` vsebuje sistemske datoteke
- `/data` vsebuje uporabniške podatke
- `/storage/emulated` vsebuje podatke glavnega pomnilnika naprave
- `/proc` vsebuje informacije o tekočih procesih

Seznam 1: Točke interesa

Preiskali in analizirali smo vse točke interesa razen mape `/proc`, saj je bila naprave do točke preiskave ugasnjena.

7.2 ANALIZA SLIK Z AUTOPSY

Vse slike smo analizirali s programskim orodjem Autopsy[1]. Autopsy je digitalna forenzična platforma ter grafični vmesnik za The Sleuth Kit in druga orodja [2]. The Sleuth Kit[2] je knjižnica in zbirka orodij, ki omogočajo analizo podatkovnih medijev [3].

Sprva smo se lotili analizirati sliko, ki vsebuje podatke predpomnilnika, t.j. `/cache`. Odrili smo le eno datoteko, ki je vsebovala uporabniška imena in gesla omrežij WIFI. Vsebnost datoteke s prekritimi gesli je prikazana na sliki 7.

```
Data
WIFI
network={
        ssid="Blazej"
        psk="[redacted]"
        key_mgmt=WPA-PSK
        priority=6
network={
        ssid="DM2010"
        psk="[redacted]"
        key_mgmt=WPA-PSK
        priority=4
network={
        ssid="eduroam"
        key_mgmt=WPA-EAP IEEE8021X
        eap=PEAP
        identity="ab4468@student.uni-lj.si"
        password="[redacted]"
        priority=3
        proactive_key_caching=1
network={
        ssid="Makar"
        psk="[redacted]"
        key_mgmt=WPA-PSK
        priority=5
network={
        ssid="Nusdorfer"
        psk="[redacted]"
        key_mgmt=WPA-PSK
        priority=7
```

Slika 7: Uporabniška imena in gesla za omrežja WIFI.

Na sistemski sliki, t.j. `/system`, nismo našli ničesar kar bi lahko povezali z uporabnikom. Tu prebivajo samo sistemske aplikacije, različne pisave in ostale sistemske datoteke.

[1] `https://www.sleuthkit.org/autopsy/`
[2] `https://www.sleuthkit.org/sleuthkit/`

Sliki `/data` in `/storage/emulated` vsebujeta največ podatkov, ki smo jih uspeli povezati z uporabnikom. Zaradi količine odkritega smo naslednji dve podpoglavji posvetili predstavitvi teh rezultatov.

7.3 AUTOPSY IN /DATA

V mapi `/data` smo našli veliko zanimivih uporabniških podatkov, ki so bili večinoma shranjeni v obliki podatkovnih baz. Pri iskanju smo si pomagali z ukazom, ki izpiše vse podatkovne baze SQLite:

```
find /data/data -name "*.db"
```

Sprva smo se pomaknili v mapo `/data/data` in pregledali podatkovno bazo `/data/data/0/com.android.providers.telephony/databases/mmssms.db`. Baza je vsebovala informacije o sporočilih SMS in MMS, kot so njihova vsebina, pošiljatelj, prejemnik, čas in druge metapodatke (slika 8). Pri pregledu SMSov smo našli tudi lokacije, kamor se shranjujejo fotografije (slika 9), poslane s sporočili MMS (`/data/user/0/com.android.providers.telephony/app_parts/`).

reply_path_present	subject	body	service_center
Filter		Filter	Filter
0	NULL	Dobro, sej ni panike.. Povej se veroniki :)	+38640441...
0	NULL	Aja lahko ti dam pol kes :)	+38640441...
0	NULL	Sej gres ti kupit ne? K sm malo tecna inje to prevel...	+38640441...
0	NULL	Ce boste naslednjo subvencijo izkoristili v katerikoli...	+38640441...
0	NULL	Aja kej gremo ob 14.30 u petek? Kej ves ce bosta ...	+38640441...
NULL	NULL	Ja ob 14:30 bo uredu in mislim da sta rekli obe da ...	NULL
0	NULL	Okej kul kul :)) bom jima napisala sms	+38640441...
NULL	NULL	Ok :)	NULL
0	NULL	Od zdej naprej ti bi mogla kartica delat ;)	+38640441...
NULL	NULL	O ful hvala k si zrihtu :)	NULL
NULL	NULL	Jst bom cez par min tam tko da ce ces lahko pride...	NULL
0	NULL	Oj, kako kej? :) kako si ti nameraval za danes potem?	+38640441...
NULL	NULL	Oj sori jst bom poznTe poklicem pole	NULL
0	NULL	Oj mi bomo pri kongresnem	+38640441...
0	NULL	Te pa lahko noc in dobro spi	+38640441...
NULL	NULL	Lahko noc :)	NULL
0	NULL	Ou pole prides?	+38640441...
0	NULL	Uzivej pr Tadeju ;)	+38640441...

Slika 8: Del podatkovne baze, ki hrani podatke o sporočilih SMS in MMS.

<image000000>	image000000.jpg	NULL	NULL	/data/user/0/com.android.providers.telephony/ap...
<text.000001>	text.000001.txt	NULL	NULL	NULL
<smil>	smil.xml	NULL	NULL	NULL
<image000000>	image000000.jpg	NULL	NULL	/data/user/0/com.android.providers.telephony/ap...
<smil>	smil.xml	NULL	NULL	NULL
<image000000>	image000000.jpg	NULL	NULL	/data/user/0/com.android.providers.telephony/ap...
<text.000001>	text.000001.txt	NULL	NULL	NULL
<i0000010.smil>	NULL	NULL	NULL	NULL
<i0000011.jpg>	i0000011.jpg	NULL	NULL	/data/user/0/com.android.providers.telephony/ap...
<el2.txt>	el2.txt	NULL	NULL	NULL
<smil>	smil.xml	NULL	NULL	NULL
<image000000>	image000000.jpg	NULL	NULL	/data/user/0/com.android.providers.telephony/ap...
<text.000001>	text.000001.txt	NULL	NULL	NULL
<i0000010.smil>	NULL	NULL	NULL	NULL
<IMG_3649.png>	IMG_3649.png	NULL	NULL	/data/user/0/com.android.providers.telephony/ap...
<smil.smil>	NULL	NULL	NULL	NULL
<text_0.txt>	text_0.txt	NULL	NULL	NULL
<20151128_1...	20151128_13...	NULL	NULL	/data/user/0/com.android.providers.telephony/ap...
<smil>	smil.xml	NULL	NULL	NULL
<image000000>	image000000.jpg	NULL	NULL	/data/user/0/com.android.providers.telephony/ap...
<text.000001>	text.000001.txt	NULL	NULL	NULL
<smil>	smil.xml	NULL	NULL	NULL

Slika 9: Lokacije in imenena slik, poslanih s sporočili MMS.

Nato smo pregledali bazo `mailstore.aljaz.blazej@gmail.com.db`, ki se je nahajala v mapi `/data/data/com.google.android.gm/databases/`. Vsebovala je informacije o poslanih in prejetih e-poštah, lokacije, kjer se nahajajo priloge in celotne vsebine pošt. Ker je bil to uporabnikov primarni e-poštni naslov, je bilo vsebine veliko.

Naslednja podatkovna baza, ki smo jo pregledali, se je nahajala na naslovu `/data/data/com.dropbox.android/app_DropboxSyncCache/hizqkiq3952astb/91130259-notifications/cache.db`. Vsebovala je obvestila, ki jih je oseba prejela v aplikaciji Dropbox. S tem smo pridobili informacije o podatkih, ki jih je oseba delila z ostalimi.

Zatem smo pregledali podatkovno bazo `/data/data/com.android.providers.contacts/databases/contacts2.db`. Vsebovala je podatke o vseh kontaktih, zgodovino klicev in za nekatere kontakte tudi sliko profila.

Našli smo tudi šifrirane javne in zasebne ključe, ki so se nahajali v datoteki `/data/data/com.google.android.gms/databases/keys.db`.

Lastnik naprave je uporabljal tudi aplikacijo Google Keep, kamor je med drugim zapisal tudi WIFI geslo. Baza aplikacije se je nahajala v `/data/data/com.google.android.keep/databases/keep.db`.

V podatkovni bazi aplikacije Skype (`/data/data/com.skype.raider/files/aljaz.blazej/main.db`) smo našli zadnjih nekaj sporočil, ki jih je uporabnik izmenjal s svojimi kontakti.

Veliko koristnih podatkov smo našli tudi v mapi brskalnika Google Chrome, ki se je nahajala na naslovu `/data/data/com.android.chrome/app_chrome/`. Spodnji seznam našteje forenzično zanimive datoteke in kaj je bilo v njih najdeno:

1. **Web Data**
   - zaznamki
   - uporabnikov naslov prebivališča
   - zadnje prebrane novice
2. **SyncData.sqlite3**
   - uporabnikov dan rojstva
3. **Cookies**
   - piškotki
4. **History**
   - zgodovina obiskanih spletnih strani
   - zgodovina ključnih besed, vnešenik v brskalnik
5. **Login Data**
   - vsa uporabniška imena in nešifrirana gesla, ki jih je uporabnik shranil
6. **Network Action Predictor**
   - dopolnjevanje teksta

7.4 AUTOPSY IN /STORAGE/EMULATED

V naslednjem koraku smo pregledali uporabniške podatke, ki se nahajajo v mapi `sdcard`. Kljub zavajajočemu imenu mape, se podatki nahajajo na notranjem pomnilniku in ne na SD kartici (Nexus 4 nima možnosti razširitve pomnilnika z SD kartico). Ti podatki so dostopni tudi brez pravic do korenskega imenika.

Sprva smo pregledali mapo `DCIM`, kjer je naprava shranjevala fotografije, ki so bile zajete z vgrajenim senzorjem. Našli smo fotografije in njihove EXIF podatke, ki so med drugim vsebovali GPS koordinate, kjer je bila fotografija zajeta.

Nato smo se pomaknili v mapo `bluetooth`, kjer se shranjujejo datoteke, ki bo bile izmenjane preko Bluetooth povezave. V mapi smo našli dokumente, ki so vsebovali srednješolsko gradivo.

Pregledali smo tudi mapo `CamScanner`, kjer smo našli fotografije zapiskov, zajete s pomočjo aplikacije *CamScanner*.

Zatem smo pogledali vsebino mape `Pictures`, kjer smo našli sličice (angl. thumbnail). Sličic je bilo bistveno več kot fotografij, najdenih v mapi `DCIM`, saj so nekatere pripadale že izbrisanim slikam.

Nato smo pregledali mapo aplikacije *Snapchat* (`Snapchat`), kjer smo našli shranjene fotografije, zajete z omenjeno aplikacijo (memories).

Med drugim smo pridobili tudi uporabnikovo glasbo (v mapi `Music`), Miniclip ID in zaslonske maske (angl. screenshot).

8. ZAKLJUČEK

V članku smo zajeli in predstavili bralcu kako pravilno zajeti, preiskati in analizirati mobilno napravo, ki poganja operacijski sistem Android. Ugotovili smo, da na Android verziji 6.0 (Marshmallow) največ podatkov povezanih z uporabnikov vsebujejo mape (i) `/data`, ki vsebuje uporabniške podatke (ii) `/storage/emulated`, ki vsebuje podatke glavnega pomnilnika naprave in (iii) `cache`, ki vsebuje predpomnilnik.

Med preiskovanjem in analizo smo se naučili veliko novih veščin in uporabili moderna orodja za preiskovanje slik kot sta Autopsy in FTK Imager. Orodji zelo pohitrita samo preiskovanje ter imata opcijo samodejnega generiranja poročil.

Nad rezultati smo bili zelo presenečeni, saj nismo pričakovali tako velikega števila podatkov, ki smo jih lahko povezali s samim uporabnikov, kot npr. gesla v golem besedilu. Po končanem pisanju pa smo poskrbeli za varnost naših pametnih naprav.

9. VIRI

[1] B. Anderson. Understanding the android file hierarchy. `http://www.all-things-android.com/content/understanding-android-file-hierarchy`. Dostopano: 12. maj 2018.

[2] B. Carrier. Autopsy. `https://www.sleuthkit.org/autopsy/`. Dostopano: 12. maj 2018.

[3] B. Carrier. The sleuth kit. `https://www.sleuthkit.org/sleuthkit/`. Dostopano: 12. maj 2018.

[4] P. R. Center. Mobile fact sheet. `http://www.pewinternet.org/fact-sheet/mobile/`. Dostopano: 8. maj 2018.

[5] G. Developers. About the platform. `https://developer.android.com/guide/platform/`. Dostopano: 11. maj 2018.

[6] J. Lessad and G. Kessler. Android forensics: Simplifying cell phone examinations. 4, 01 2013.

[7] StatCounter. Mobile operating system market share worldwide. `http://gs.statcounter.com/os-market-share/mobile/worldwide/`. Dostopano: 8. maj 2018.

[8] S.-T. Sun, A. Cuadros, and K. Beznosov. Android rooting: Methods, detection, and evasion. In *Proceedings of the 5th Annual ACM CCS Workshop on Security and Privacy in Smartphones and Mobile Devices*, SPSM '15, pages 3–14, New York, NY, USA, 2015. ACM.

[9] Wikipedia. Android (operating system). `https://en.wikipedia.org/wiki/Android_(operating_system)`. Dostopano: 11. maj 2018.

Ocena integritete podatkov pri mobilni forenziki s spremljanjem dogodkov*

[Računalniška forenzika 2017/2018]

Jernej Janež
Univerza v Ljubljani
Fakulteta za računalništvo in informatiko
Večna pot 113
1000 Ljubljana, Slovenija
jj5715@student.uni-lj.si

Vitjan Zavrtanik
Univerza v Ljubljani
Fakulteta za računalništvo in informatiko
Večna pot 113
1000 Ljubljana, Slovenija
vz1528@student.uni-lj.si

POVZETEK

Zaradi širokega spektra storitev, ki jih mobilni telefoni ponujajo, postajajo vedno bolj pomembno orodje vsakdanjega življenja ljudi. Zato bi lahko delovali kot temeljne priče ali preprosto kot vir informacij pri podpiranju preiskav številnih kaznivih dejanj, ki niso omejena le na digitalni kriminal. Trenutna forenzična pridobitev in analiza naprav temelji na orodjih za izvajanje daljinskega upravljanja, pri katerih se uporablja forenzično delovno postajo za zaseg z vstavljanjem kode v mobilno napravo. Iz tega sledi, da je karakterizacija spoštovanja integritete še vedno težavna in potrebuje poglobljeno raziskavo. Avtorji članka [5] so predstavili nov pristop za oceno spoštovanja integritete v zvezi z orodji za pridobivanje informacij. Rezultati poskusov kažejo primernost predlagane strategije.

Ključne besede

mobilna forenzika, spoštovanje integritete, ocena korupcije

1. UVOD

Mobilni telefoni trenutno predstavljajo eno izmed najbolj razpršenih tehnologij po vsem svetu [9]. Trenutne zmogljivosti mobilnih telefonov so izredno zanimive iz vidika forenzičnih preiskav [8]. Še več, mobilni telefoni so opremljeni z nizom vmesnikov, ki omogočajo tako dolge (npr. GSM, UMTS) kot kratke (npr. Bluetooth, WiFi) razdalje komunikacij. Glede zmogljivosti so bolj podobni računalnikom, kot pa le telefonom. Posledično uporaba teh funkcionalnosti povečuje količino in kakovost osebnih podatkov, shranjenih v mobilnih telefonih. Pravzaprav informacije shranjene na mobilnih telefonih, natančno opisujejo navade in vedenja njihovih lastnikov.

Število mobilnih telefonov vpletenih v kriminalnih dejanjih zmeraj bolj narašča, kar je privedlo do večje potrebe forenzičnih analiz mobilnih telefonov. Avtorji v tem prispevku predstavljajo novo strategijo za izvedbo boljše ocene spoštovanja integritete od pristopa uporabljenega v [6]. Ta pristop uporablja aplikacijo, ki je bila posebej zasnovana za zajem in prepoznavanje morebitnih sprememb, ki so se zgodile v datotečnem sistemu mobilnih telefonov, s posebno skrbjo za notranji pomnilnik. Na ta način je mogoče pravilno identificirati vsako spremembo in z njo povezani subjekt.

*Članek je povzet po [5].

2. PREGLED PODROčJA

Danes je pogostost sistema Symbian zanemarljiva v primerjavi s sistemom Android ali iOS [1], ki skupaj obvladujeta več kot 90% tržni delež. Kljub temu, da se današnje naprave bistveno razlikujejo od mobilnih telefonov, ki so poganjali Symbian, pa se težave pri forenzični obdelavi niso bistveno spremenile. Še vedno je zaradi stalne povezanosti sistemov, težko zagotoviti integriteto pridobljenih podatkov. Potreba po metodah za ugotavljanje učinkovitosti novo razvitih forenzičnih orodji, pri ohranjanju integritete podatkov narašča s številom uporabnikov mobilnih telefonov. Poleg naraščajočega števila uporabnikov, pa se je povečala tudi količina podatkov, ki jih uporabniki hranijo na mobilnih telefonih.

Pridobivanje podatkov iz mobilnih naprav je težko izvesti brez kakršnekoli izgube podatkov na sami napravi. Nalaganje ali uporabljanje aplikacij, prižiganje ali ugašanje naprave in širok nabor drugih interakcij z napravo lahko povzroči izbris ali prepis datotek, ki bi lahko služile kot dokazno gradivo. Veliko podatkov lahko dobimo že z ročnim pregledovanjem naprave, vendar pa je tveganje za uničevanje pomembnega gradiva na napravi zelo visoko. Bolj kontrolirano je dostopanje do datotečnega sistema ali pa preko kopiranja samega diska, kot lahko storimo z JTAG vmesnikom [2]. Za fizični dostop do diska je včasih potrebno napravo delno razstaviti, kar je lahko problematično za ohranjanje dokazov, vendar pa so postopki kot je branje preko JTAG vmesnika sprejemljivi, ker imajo možnost odkritja izbrisanih datotek in ker je v nekaterih primerih to najboljša opcija [2].

Pogosto se do datotečnega sistema dostopa preko načina za povrnitev podatkov (Recovery mode), vendar pa je za to pogosto potreben ponovni zagon naprave in pa doseganje administratorskih privilegijev na napravi, kar pa zahteva spremembe v sistemskih datotekah ali pa izkoriščanje varno-

stnih lukenj sistema. V vsakem primeru je težko zagotoviti integriteto podatkov.Pri pridobivanju podatkov je problematično tudi obhajanje raznih varnostnih mehanizmov, ki vključujejo enkripcijo podatkov kot so enkripcija diska FDE (Full Disk Encryption), enkripcija FBE (File-based Encryption), varno nalaganje (Secure Boot) in enkripcija podatkov posameznih aplikacij [12].

V delu [11] avtorji predlagajo sistem za ekstrakcijo podatkov iz mobilnih naprav z operacijskim sistemom Android, integriteto podatkov po uporabi njihovega sistema pa preverjajo s predhodno pridobljenimi podatki preko JTAG vmesnika.

V delu [12] predstavijo še orodje za pridobivanje podatkov iz glavnega spomina s preklopom v način za posodobitev strojne programske opreme (Firmware update mode), ki omogoči dostop do glavnega spomina brez ponovnega zagona naprave. Preklop v ta način v nekaterih verzijah Android sistema ne potrebuje administratorskih pravic ali odklenjenega ekrana, za kar je potrebno velikokrat posegati v sistemske datoteke, s čemer pa tvegamo izgubo integritete podatkov. Poleg tega, so v glavnem spominu pogosto shranjeni ključi pomembni za dekripcijo ostalih podatkov na napravi v primeru uporabe varnostnih orodij kot so enkripcija diska FDE (Full Disk Encryption) [12], ki se vedno pogosteje uporabljajo na mobilnih napravah.

3. OCENA INTEGRITETE

Čeprav je NIST šele maja 2007 objavil dokument [8], ki zajema področje forenzičnih preiskav mobilnih telefonov, je bilo v preteklosti veliko truda, da bi ustvarili niz pravil in smernic, ki bi opisale dobre prakse upravljanja s podatki med preiskavami. Pravzaprav zaradi nedostopnosti notranjega pomnilnika, klasična pravila in smernice računalniške forenzike ne morejo biti zgolj preusmerjene na mobilne naprave; verjetno glavni zaviralec predstavlja nerazpoložljivost/heterogenost neposrednega dostopa do vseh podatkov [7]. Vendar, če ta pravila in smernice pomagajo pri preprečevanju korupcije datotečnega sistema, so šibke pri opredeljevanju in preučevanje tega pojava. Analitični potek dela je močno odvisen od začetnega stanja naprave; v takih scenarijih se minimalni stopnji korupcije zdi nemogoče izogniti. Zato je trenutni cilj pri mobilni forenziki zmanjšati stopnjo korupcije shranjenih podatkov, pri čemer je osnovni predlagani pristop, globoko analizirati korupcijo, z namenom ločiti izvirno informacijo od zastrupljene.

Ker je trenutno validacija forenzičnih orodij draga in zapletena naloga, se veliko proizvajalcev bolj zanima za funkcionalnost, kot pa za močno potrditev njihovih forenzičnih orodij [7].

3.1 Ocena integritete z branjem kode

Branje kode se pogosto uporablja za izvajanje generičnega testiranja. Ta tehnika je lahko uporabna tudi pri ocenjevanju forenzičnega orodja, ki zagotavlja podrobno analizo obnašanja orodja. Vendar ta pristop vsebuje tri glavne težave:

- Orodje mora biti odprtokodno: za druga orodja, kjer izvorna koda ni na voljo, je nemogoče izvesti pregled kode [3];
- Celotno operativno okolje mora biti odprtokodno: ker je forenzično orodje le del celotne opreme, je pomembno razširiti pregled kode na celotno okolje;
- Sklepi temeljijo zgolj na statičnem razumevanju vedenja orodja in ne morejo upoštevati dinamičnih nalog.

Na žalost je ta pristop pri mobilni forenziki izjemno zapleten in se ga v praksi manj uporablja, saj je celotno delovno okolje redkokdaj popolnoma odprtokodno.

3.2 Ocena integritete s pomočjo eksperimentiranja

Naslednja strategija opravi oceno v obratni smeri: samo dinamični razvoj orodja in rezultati se uporabljajo. Ta način se pogosteje uporablja v praksi in temelji na zmožnosti izvleči referenčno sliko podatkov, ki jih je treba ohraniti. Izdelana je tudi dodatna slika teh podatkov in nato prečno preverjanje med dvema slikama, da bi identificirali razlike. Ta pristop predstavlja veliko več praktičnih koristi glede na prvega, vendar zmanjšana velikost in reprezentativnost vzorca, bi lahko ogrozila veljavnost in posplošitev sklepov.

Poleg tega ni vedno mogoče enostavno dobiti zahtevano referenčno sliko in v mnogih primerih je edino orodje, ki lahko dobi referenčno sliko, prav to orodje, ki ga preiskujemo ali njemu podobno. V tej situaciji je težko sklepati kakšno vrsto in stopnjo korupcije povzroča orodje oz. drugi dogodki.

4. MIAT ORODJE

Delo opisano v [4, 6, 10], predstavlja novo metodologija za pridobivanje podatkov iz pametnih telefonov, ki temelji na lokalni povezavi med mobilno napravo, ki postane forenzična delovna postaja z notranjim pomnilnikom telefona. Forenzično orodje, na katerem temelji nova metodologija imenovanim MIAT (Mobile Internal Acquisition Tool) je programska oprema, ki se lahko namesti neposredno na pametni telefon s pomočjo pomnilniške kartice. Med izvajanjem MIAT zrcali notranji pomnilnik na pomnilniško kartico na tak način, da celotna forenzična oprema, potrebna za opravljanje zajema, je nabor pomnilniških kartic.

Ker je MIAT namenjen kot forenzično orodje, je ocena forenzičnih lastnosti potrebna. V [6] je dana celovita ocena tako za zmogljivost kot za forenzične lastnosti. Vendar pa ta pristop ne zahteva oceniti, ali je korupcija posledica postopka pridobitve podatkov ali kakšnega drugega vzroka.

Iz tega razloga so avtorji tega prispevka predstavili močnejšo analizo pojava korupcije; nova strategija omogoča ločevanje med izdelavo referenčne slike in zajemom z orodjem, ki se ga preiskuje.

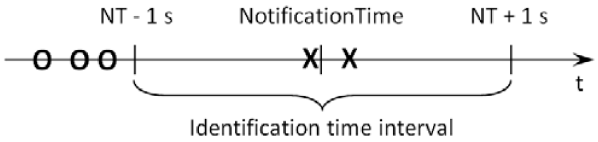
5. PREDLAGANA STRATEGIJA

Strategija predstavljena v prispevku [5], temelji na aplikaciji imenovani FSMon, ki je zasnovana in implementirana za izkoriščanje API-jev Symbian operacijskega sistema. FSMon lahko hitro ustvari sliko celotne strukture notranjega pomnilnika, ter ima možnost zaznati kakršnekoli spremembe na sistemu.

5.1 FSMon aplikacija

Glavne naloge aplikacije so zaznati operacije pisanja, ustvarjanja, ter brisanja datotek, ki se zgodijo na datotečnem sistemu notranjega pomnilnika. Iz tega sledi, da se FSMon lahko izvede na 2 različna načina:

1. *Čakanje obvestil*: ko se FSMon vzpostavi na način čakanja obvestil uporablja neskončno zanko. V vsaki ponovitvi čakamo na obvestilo in ko se ta zgodi, shranimo čase zadnjih sprememb podatkov za vsak vnos, ki je trenutno prisoten v datotečnem sistemu. Med shranjenimi podatki FSMon išče vnose, ki so bili spremenjeni, ustvarjeni ali izbrisani v omejenem časovnem razponu v času obvestila, da bi prepoznali tiste vnose datotečnega sistema, na katere je vplivalo zadnje obvestilo. Na sliki 1 je prikazan primer časovnega razpona dveh sekund. Ta pregled je primeren samo za zaprte datoteke: ob času obvestila, če je izbrana datoteka še vedno odprta, je potrebno uporabiti drugačen pristop. V tem primeru, se velikost datoteke primerja s prejšnjo; če pride do neujemanja, je datoteka potrjena.



Slika 1: Primer časovnega razpona (Identification time interval). Vnosi z zadnjimi spremembami v časovnem razponu (X) so izbrani, drugi (O) pa se zavržejo.

2. *Kreiranje slike datotečnega sistema*: ko se FSMon vzpostavi na ta način, ustvari sliko drevesa datotečnega sistema, vključno s časi zadnjih sprememb za vsak vnos. Ta način je uporaben pri raziskovanju korupcije pri dogodkih, ki jih FSMon ne more nadzirati (npr. ponovni zagon naprave).

V obeh načinih so rezultati shranjeni na isto pomnilniško kartico, ki se uporablja za shranjevanje FSMon aplikacije, ter enostavne besedilne datoteke. V prvem načinu ta besedilna datoteka vsebuje vnose datotek, na katere vplivajo obvestila, v drugem načinu pa datoteka vsebuje ustvarjeno sliko.

6. OPREDELITEV POSKUSOV

Pri izvajanju poskusov so avtorji članka [5] opredelili naslednje definicije.

6.1 Nadzor stanja naprave

Najprej moramo nadzorovati stanje uporabljene mobilne naprave; stopnjo nadzora, ki jo potrebujemo, mora biti zagotovljena, da se vsak poskus začne in nadaljuje pod stalnimi razmerami. Stanje naprave je nadzorovano z uporabo štirih protiukrepov:

1. Napravo je potrebno ponastaviti pred vsakim poskusom, da zagotovimo, da poskusi delujejo na isti sistemski sliki;
2. Naprava se vedno zažene v načinu za obnovitev, da zagotovimo, da se zaženejo le ključne aplikacije po zagonu naprave, medtem ko je izvajanje drugih aplikacij onemogočeno; to preprečuje korupcijo zaradi nekaterih lokalnih dogodkov (npr. aplikacije ki se zaženejo ob zagonu naprave);
3. Naprava se zažene v načinu brez povezave, tako da odstranimo SIM kartico in s tem zagotovimo izolacijo med mobilno napravo in komunikacijskim omrežjem; dogodki (npr. dohodni klici ali besedilna sporočila) ne morejo pokvariti shranjenih podatkov;
4. Raven napolnjenosti baterije je najmanj 50% in polnilec ni priključen.

6.2 Subjekt poskusov

Vsi poskusi so bili izvedeni na napravi Nokia N70, na kateri je bil naložen Symbian OS v8.1a.

6.3 Ponovljivost poskusov

Ponovljivost poskusov je eden od ključnih vidikov forenzičnega raziskovanja; na splošno, replikacijo poskusov je močno povezana z veljavnostjo in posplošitvijo rezultatov.

6.4 Eksperimentalni potek dela

Avtorji so opredelili tri poteke dela:

(a) *Karakterizacija forenzičnega vedenja orodja*: podatke zbere aplikacija FSMon in nato se izvede prečno preverjanje pridobljenih podatkov.

(b) *Karakterizacija preprostih dogodkov*: preiskovanje korupcije datotečnega sistema zaradi pojava dogodkov, ki jih FSMon lahko nadzira.

(c) *Karakterizacija kompleksnih dogodkov*: ta potek dela ima enak cilj kot prejšnji, a nekaterih dogodkov (npr. ponovni zagon naprave, odstranitev SIM kartice) aplikacija FSMon ne more nadzorovati.

7. REZULTATI POSKUSOV

7.1 Pridobitev podatkov s pomočjo MIAT orodja

Potek dela, opisan v poglavju 6.4(a) so avtorji uporabili dvakrat, da bi zbrali tako operacije pisanja, kot tudi ustvarjanja in brisanja datotek. FSMon je našel 3 datoteke, ki so bile spremenjene med procesom zajema; vendar se dve datoteki spremenita šele po tem, ko se zajem zaključi. Z drugimi besedami, MIAT sliki teh dveh datotek vsebujeta iste podatke, kot jih hranita originalni datoteki pred pridobitvijo.

7.2 Dogodki naprave

(a) *Ponovni zagon naprave*: potek dela, opisan v poglavju 6.4(c) so uporabili enkrat. V primerjavi med zagonom po ponastavitvi naprave z nadaljnjimi zagoni lahko ugotovimo, da na stopnjo korupcije zaradi ponovnega zagona naprave vpliva prejšnji postopek ponastavitve naprave. Izkaže se, da se med zagonom po ponastavitvi naprave ustvari 1 nova datoteka, ter 16 zapisov v datoteke. Z nadaljnjimi zagoni se pa izvede samo operacija pisanja na 14 datotekah.

(b) *Odstranjevanje SIM kartice*: potek dela, opisan v poglavju 6.4(c) so uporabili enkrat. V primerjava med odstranitvijo SIM kartice po ponastavitvi naprave z nadaljnjim odstranjevanjem lahko ugotovimo, da na stopnjo korupcije zaradi odstranitve SIM kartice iz naprave vpliva prejšnji postopek ponastavitve naprave. Podobno kot pri ponovnem zagonu se med odstranitvijo SIM kartice po ponastavitvi naprave ustvarita 2 novi datoteki, ter 21 zapisov v datoteke. Z nadaljnjimi odstranitvami SIM kartice se pa izvede samo operacija pisanja na 16 datotekah.

(c) *Namestitev MIAT orodja*: za pridobivanje podatkov z orodjem MIAT, mora upravljavec najprej orodje namestiti; zato moramo tudi raziskati stopnjo povzročene korupcije ustvarjeno z namestitvijo. Potek dela, opisan v poglavju 6.4(b) so uporabili dvakrat, da bi zbrali tako operacije pisanja, kot tudi ustvarjanja in brisanja datotek. V tem primeru ne moremo uporabiti prečnega preverjanja, saj orodja ni mogoče uporabiti brez namestitve. Opazimo, da ni razlike med prvo namestitvijo in nadaljnjimi, saj je v vseh primerih enako število operacij.

(d) *Vstavitev/odstranitev pomnilniške kartice*: za pridobivanje podatkov z orodjem MIAT, je potrebna zamenjava pomnilniške kartice; zato so avtorji morali raziskati stopnjo povzročene korupcije ustvarjeno s tem dejanjem. Potek dela, opisan v poglavju 6.4(b) so uporabili dvakrat, da bi zbrali tako operacije pisanja, kot tudi ustvarjanja in brisanja datotek. Opazimo, da zamenjava pomnilniške kartice vpliva na enak način, kot preproste operacije vstavljanja in odstranjevanja podatkov.

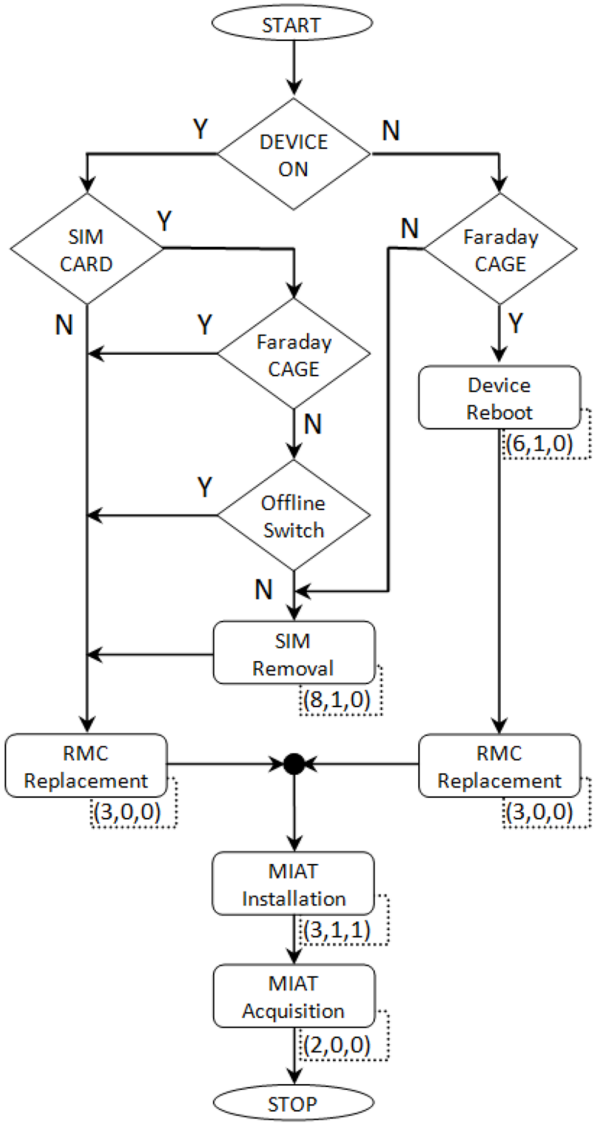
(e) *Dogodki, povezani s pridobivanjem podatkov*: ta del izpostavlja rezultate, povezane z nekaterimi dogodki, ki jih je mogoče povezati z zbiranjem podatkov; potek dela opisanega v poglavju 6.4(b) so za vsako od teh uporabili dvakrat, da bi zbrali tako operacije pisanja, kot tudi ustvarjanja in brisanja datotek. Zlasti so se osredotočili na naslednje tri dogodke:

- *Stikalo za mobilne podatke*: da bi pridobili podatke z uporabo MIAT orodja, naj bi bil telefon ločen od omrežja; to lahko dosežemo s programskim stikalom.
- *Priključek polnilca*: pridobivanje podatkov lahko zahteva veliko časa ne glede na uporabljena orodja. Da bi preprečili ustavitev procesa zaradi prazne baterije, se naprava lahko priključi na polnilec.
- *Priključek USB kabla*: za pridobitev podatkov iz telefona, bi se ta lahko povezal s forenzično delovno postajo z USB kablom.

Poskusi kažejo, da ni bila nobena datoteka spremenjena, ne ustvarjena in ne izbrisana pri vseh zgornjih primerih.

8. CELOTEN POTEK ZAJEMA PODATKOV S POMOČJO MIAT ORODJA

Celoten potek procesa pridobitve podatkov s pomočjo MIAT orodja je mogoče označiti z vidika spoštovanja integritete, z uporabo rezultati iz poglavja 7. Cilj avtorjev je bil podati opis, kako lahko dogodki med seboj interoperirajo v vseh možnih pretokih dogodkov. Ker stanje naprave, ki jo je treba pridobiti in opreme na kraju zločina ni mogoče predčasno poznati, so združili vse možne pretoke v enoten diagram poteka prikazan na sliki 2. Slika prikazuje takšen diagram, kjer je vsaka pot od začetka do končnega stanja predstavlja popoln operativni pretok. Pričakovano korupcijo celotnega pretoka lahko izračunamo kot unijo vsake dimenzije dogodkov na poti.



Slika 2: Ocena korupcije za celoten potek zajema podatkov s pomočjo MIAT orodja. Vsak funkcionalni korak je opisan s tremi dimenzijami, ki merijo število datotek, ki so bile spremenjene, ustvarjene in izbrisane v danem koraku.

9. ZAKLJUčEK

Dobra ocena lastnosti mobilnih forenzičnih orodij je težka naloga in pogosto bolj redko izveden korak med oblikovanjem in implementacijo orodij. Vendar odprtokodna programska oprema in skupnosti spodbujajo globoko testiranje in ocenjevanje aplikacij za splošne namene. Korupcije podatkov mobilnih naprav, se je težko izogniti in trenutno realistični cilj bi lahko bil minimizirati, kot tudi opredeliti značilnosti korupcije med celotno preiskavo.

Avtorji članka [5] so predstavili novo strategijo za karakterizacijo korupcije notranjega pomnilnika zaradi forenzičnih orodij in nastajajočih dogodkov. Nadaljnji poskusi so omogočili karakterizacijo celotnega procesa zajema podatkov s pomočjo MIAT orodja; takšna opredelitev bi lahko pripomogla forenzičnim preiskovalcem k boljšem upravljanju naprav odkritih na krajih zločina, da bi zmanjšali stopnjo korupcije.

10. LITERATURA

[1] Mobile phone os market share. `https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/`. Accessed: 2018-05-13.

[2] K. Barmpatsalou, D. Damopoulos, G. Kambourakis, and V. Katos. A critical review of 7 years of mobile device forensics. *Digital Investigation*, 10(4):323–349, 2013.

[3] B. Carrier. Open source digital forensics tool. *The Legal Argument*, 2003.

[4] F. Dellutri, V. Ottaviani, and G. Me. Miat-wm5: forensic acquisition for windows mobile pocketpc. In *Proceedings of the workshop on security and high performance computing, as part of the 2008 international conference on high performance computing & simulation*, pages 200–5. Citeseer, 2008.

[5] A. Distefano, A. Grillo, A. Lentini, G. Me, and D. Tulimiero. Mobile forensics data integrity assessment by event monitoring. *Small Scale Digital Device Forensic Journal (SSDDFJ)*, 468, 2010.

[6] A. Distefano and G. Me. An overall assessment of mobile internal acquisition tool. *digital investigation*, 5:S121–S127, 2008.

[7] W. Jansen, A. Delaitre, and L. Moenner. Overcoming impediments to cell phone forensics. In *Hawaii International Conference on System Sciences, Proceedings of the 41st Annual*, pages 483–483. IEEE, 2008.

[8] W. A. Jansen and R. Ayers. *Guidelines on Cell Phone Forensics: Recommendations of the National Institute of Standards and Technology*. US Department of Commerce, Technology Administration, National Institute of Standards and Technology, 2006.

[9] K. Kalba. The adoption of mobile phones in emerging markets: Global diffusion and the rural challenge. *International journal of Communication*, 2:631–661, 2008.

[10] G. Me and M. Rossi. Internal forensic acquisition for mobile equipments. In *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*, pages 1–7. IEEE, 2008.

[11] N. Son, Y. Lee, D. Kim, J. I. James, S. Lee, and K. Lee. A study of user data integrity during acquisition of android devices. *Digital Investigation*, 10:S3–S11, 2013.

[12] S. J. Yang, J. H. Choi, K. B. Kim, R. Bhatia, B. Saltaformaggio, and D. Xu. Live acquisition of main memory data from android smartphones and smartwatches. *Digital Investigation*, 23:50–62, 2017.

iPhone Forensics

Aleksandra Turanjanin and Katarina Milačić
Faculty of Computer and Information Science
University of Ljubljana
Ljubljana, Slovenia

ABSTRACT

iPhone is one of the most popular mobile devices today and therefore it is logical that it can represent the essential part in an investigation, since vital information from these devices can make critical part of investigative evidences. The challenge is the extraction of data of forensic value such as e-mail messages, text and multimedia messages, calendar events, browsing history, GPRS locations, contacts, call history, voicemail recording, etc.

1. INTRODUCTION

Nowadays smartphones have become important part of life. Among other devices, iPhone turned out to be the most popular choice recently. Apple earned almost 80% of the worldwide profit on smartphones in 2016. Till today 1.16 billion of iPhone devices has been sold [1].
Since these devices can represent essential source of data necessary in investigation, various methods have been developed for acquiring important information. This paper first introduces organisation of data in iOS devices. Later methods for data acquisitions are explained, with special attention to logical extraction and forensic tools, as well as iOS backup options. When performing examination of iOS devices, important artifacts such as messages, contacts, GPRS locations, call logs, photos, calendar events etc. are extracted and analyzed.

2. STRUCTURE OF DATA

In order to gain better insight in iOS forensics, here is presented data structure in iOS devices, such as division of partitions, file system, memory and storage organization.

2.1 Flash Memory

In order to maintain small physical memory, iPhone uses flash memory[14] that doesn't need power to maintain data on the chip. The iPhone contains NAND chip, that also needs a RAM memory to work.
Flash memory has a more limited lifetime compared to other hard drives because of the "wearing" that erasing data does to the chip. Before the chip gets wearied out, there is limited number of times erasing can be done.
Flash memory can present problem for forensic investigation, since the memory shifts data around and overwrite sectors or pages without the operating system controlling it. This leads to unpredictability, and we want to keep the data unchanged when acquiring a memory.

2.2 Partitions

Partitions on an iOS device are divided into the system partition and the data partition [14]. The first one contains the software for running the iOS device, and the second one all the files and data used by the iOS device user.
The system partition is a read-only partition but firmware updates can be done on it. iTunes is responsible for overwriting the partition with a brand-new partition when performing an upgrade to the system. The size of this partition varies between 0.9 to 2.7 GB, and the exact size is determined by the size of the NAND driver. This does not contain any user data but upgrades files, system files and basic applications.
The data partition contains the user data.This is where the entire iTunes applications and the profile data of the user is held. When performing an investigation process, this partition is critical for collecting evidence.

Figure 1: Partitions in iOS device[14]

As it can be seen in Fig.1, iPhone has a single disk hence it is denoted as Disk0. The system partition is Disk0s1, and Data Partition is Disk0s2.

2.3 The iOS Storage with HFS File System

In order to support the large storage need of iOS, Apple introduced a new file system designed specifically for this operating system - HFS (Hierarchical File System)[10]. The Hierarchical File System (HFS) is formatted with a 512 byte block scheme and it uses B-trees (Balanced tree) to organize

data. Trees are consisting of nodes.

There are two types of blocks in the HFS system: logical blocks and allocation blocks.
The logical blocks are formatted with 512-byte block scheme. They are numbered from the first to the last block present on the given volume and they remain static.
The allocation blocks are the groups of logical blocks that are tied together as groups in order to increase the performance of HFS.

The iOS file system consists of the following[14] [Fig.2]:

- Boot Blocks - The first 1024 bytes in the sector 0 and 1 are known as boot blocks.
- Volume Header - The next 1024 bytes after the boot blocks are the volume header of HFS, which contains the information of the entire volume. The last 1024 bytes of the volume are occupied by the backup of the volume header (Alternate Volume Header).
  First there is the volume signature of the file system with the value of "HX", then version and an attributes field. The header also contains fields about volume like createDate, modifyDate, backupDate and fileCount, lastMountedVersion, etc.
- Allocation File - It tracks the allocation blocks that are currently in use by the system and the ones that are free. The size of the allocation file can be changed. It basically includes a bitmap. Each bit represents the status of the allocation block. If it is set to 1, that means that the allocation block is used, and if 0, that it is not used.
- Extent Overflow File - This file tracks the allocation tables that are used by the file. This information is recorded in a proper order in the form of balanced tree format. When the list of disk segments gets too large, some of those segments (or extents) are stored on disk in a file called the extents overflow file.
- Catalog File - The HFS uses catalog files in order to describe the files and folders present in the volume. It organizes data using balanced tree system. In iOS forensic analysis, it maintains the hierarchy of nodes like header, leaf, index, and map. In addition to this, it also contains the metadata of the files like created, modified and accessed dates.
- Attribute File - It contains the customizable attributes of a file.
- Startup File - It assists the booting system which does not have built-in ROM support.

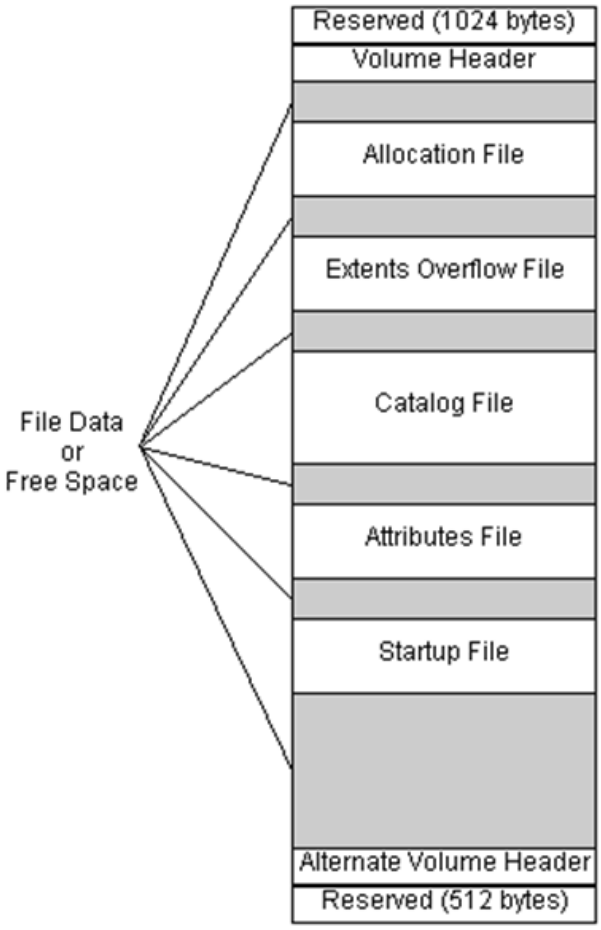


Figure 2: File Data[14]

2.4 Databases and plists

Data in the iPhone is stored in two ways, in SQLite databases [4][8] and in binary lists called "property lists" (.plist) [4]. SQLite file format is the most popular format for open source applications as well as phones. The applications which make use of SQLite database are Calendar, Messages, Notes, Address Book, and Photos. Call history, messages, geo-locations and keychains are examples of data that is stored in the databases. To read this data SQLite viewer is needed.
The Property List (plist) is a data file that is used to store data in the iOS operating system. At the very beginnings, NeXSTEP and binary formats were utilized, but nowadays the formats which can be found are either an XML format or a binary format. These lists typically contain configurations and preferences. Examples of the data which use plist file formats is our browsing history, favorites, configuration data, and others. There is a chance that plists can be opened using a standard text editor, and if not, particular tool which depends mainly on a command line interface can be used, such as tool plutil.

2.5 Backup

A backup[16] is a copy of data from database that can be used to reconstruct that data. Backups can be divided into physical backups and logical backups.
Physical backups are backups of the physical files used in storing and recovering database, such as datafiles, control files, and archived redo logs. Ultimately, every physical backup is a copy of files storing database information to some other location. Physical backups are the foundation of any sound backup and recovery strategy.
Logical backups contain logical data exported from a database with an Oracle export utility and stored in a binary file, for later re-importing into a database using the corresponding Oracle import utility. Logical backups are a useful supplement to physical backups in many circumstances, but they are not sufficient protection against data loss without physical backups.

3. ACQUISITIONS

Acquisition[3] can be considered the most important task during the investigative process. The iOS device relies on flash memory rather than a hard disk. In the iOS environment, full acquisition becomes difficult to achieve, as there is a need for the investigator to interact with several processing layers: the hardware layer, the OEM (Original Equipment Manufacturer) layer and the application layer. The hardware layer includes the processor, RAM, ROM, antenna, and other input/output devices. The OEM layer maintains the boot loading, configuration files and the application layers. Finally, the application layer supports the end user applications, internet applications, remote wiping and media players.
Here are presented six methods of acquisition: manual, logical, hex-dump analysis, chip-off, backup analysis and bit-by-bit. Depending of information that an examiner needs to find, he would use some of the following acquisition methods.

3.1 iOS Backup

Much valuable information can be found from the iOS backup [5][11]. Users have two options to backup their data. One is using Apple iTunes software, and another is Apple cloud storage service known as iCloud. Every time phone is connected to the iCloud or computer it creates the backup by copying files from the device. What to include in backup can be determined by the user. Data retrieved from iCloud or iTunes may differ.

3.1.1 iTunes

In situations when iPhone device is not available, it is common to analyze the latest backup of the device. The backup is retrieved from the device iPhone connects to for updates and syncing music, applications etc. During updating and syncing iTunes[10][11] performs an automated backup. The root of the backup folder contains the status, info and manifest plist files. The status.plist provides data about the latest backup. The info.plist file contains data that can be used to confirm the backup matches the device. The IMEI number can be found here along with the phone number. The backup files themselves which are binary in nature are converted to a SHA1 hash value of the original filename. To view these files they must be converted to a legible, human readable format. The *.mddata and *.mdinfo files are the binary files that contain the user data and will be the most interesting.

iPhone backup is created using free utility available for Mac and Windows platforms. It uses proprietor protocol to copy data from iOS device to a computer. Using a cable or Wi-Fi, iPhone can be synced with a computer. There is an option to create encrypted backup, but by default, it creates an unencrypted backup. Addition access to the data stored in iOS can be accessed when encrypted backup is cracked. To search for the information either we could create a fresh backup, or we could extract data from the existing iOS backup file. To uncover artifacts examiner needs to forensically analyze each backup.

Synchronization process gets automatically initiated once iOS device is connected to the computer. Sets of pairing records are exchanged between the iOS device and computer when iTunes detects the iOS device. Using pairing mechanism computer establishes trusted relation with iPhone. Personal information on iDevice or backup can be initiated once the computer is paired. Starting from iOS 7 pairing mechanism has been introduced.

/var/root/Library/Lockdown/pair_records/ contains pairing records. Multiple pairing records are contained if the device is paired with multiple computers. These records are stored as a property list (.plist) files. HOST ID, root certificate, device certificate and host certificate are contained within these plist files.

3.1.2 iCloud

Apple provides cloud computing service as iCloud storage [9][11][13]. Data like documents, bookmarks, calendars, contacts, photos, reminders, applications and more can be kept using the iCloud account in cloud storage. Automatically and wirelessly users can backup their iOS devices to iCloud.

5GB free storage is available with the sign-up. iCloud backup can be turned on by navigating settings. Data on the phone can be backed up automatically when the phone is plugged in, locked, and connected to Wi-Fi. As long as space is available to create current backup, iCloud provides a real-time copy of data available on the phone.

Apple's UserID and password must be known to extract backup from iCloud. Using Apple ID and password user can log on to the iCloud website and can access contacts, e-mail, calendar, photos, reminders and so on. Using Elcomsoft Phone Breaker[17], one can extract iCloud backup. If one does not have username and password of the Apple ID, iCloud backup can be extracted using the Binary token available on the computer which was synced to iCloud. Using Authentication token user can bypass login of Apple iCloud as well as bypass two - factor authentication if set by the user.

3.2 Logical Methods

Logical mobile phone acquisition systems[3][10] interact with the phone operating system to extract data. This is the most popular approach today and many tool sets have been developed for this method. The allocated, active files on the iOS

device are recovered and analyzed using the synchronization method, and data like SMS, call logs, contacts, email accounts, calendar events, web history, photos are gathered. However, this method has some limitations since it cannot extract data in slack space and recover deleted items. In this situation a physical acquisition is required.
Following tools are mostly used ones for acquiring data form iOS devices. The data retrived using these tools is very similar, but there are limitations for using

3.2.1 Lantern 4

The Lantern forensics suite developed by Katana Forensics INC [7] was designed to physically extract an image of the iOS device. Lantern imager can both decrypt the image and brute force a simple passcode (4 digits) along with providing a SHA-1 hash value. Latern can quickly extract, report, and share vital data from Apple iOS, unlock the Address Book to discover personal relationships, explore the history and duration of placed and received calls, read and archive SMS and iMessage conversaton history. All of this together with previous case files, iCloud and computer backups can be acquired into one consolidated case file. Just doing that will automatically trigger Link Analysis. In that way it is automatically available to see who is communicating to whom. Link Analysis was designed to be intuitive and uncomplicated in order to decipher thousands of pieces of information. A file system viewer integrated into the application itself for manual analysis with a built in plist editor is also present in Latern 4. It supports iOS11, decrypts encrypted backups with known password, has date filter on Data and Reports, has ability of acquisition multiple device within one case file a but all of this is only available for Intel Based Mac computers.

3.2.2 iXAM

iXAM[6] is designed to deliver evidence to a law enforcement investigation, providing anything from a stored contact or text message to an email, photograph or specific map location. The forensics' tool read is a byte level physical data copy which can be set to target specific data sets or the entire file system. iXAM does not modify the NAND flash and does not apply kernel patches used in jail breaking techniques. When used in forensic imaging mode, the output from iXAM is a raw disk image file in Apple's proprietary DMG format.

3.3 Jailbreaking

Jailbreaking[3] a phone is a technique used for replacing the firmware partition with a hacked version that will allow the examiner to install tools that would not normally be on the device. By jailbreaking a device, the current limitations of iTunes can be subverted and root access achieved. If the examiner performs this, he can use tool such as SSH and Terminal that normally are not available in order to produce a full drive image extraction.
To obtain an image of the partition the iOS device must be jailbroken. One of the methods for jailbreaking is with redSn0w. The redSn0w tool has a simple wizard that will step the iOS device through the process of replacing the firmware and installing the Cydia application. Once the device has completed the process, the examiner can begin to extract artifacts.
To begin an extraction of the iOS device image, the forensic workstation would be placed on the same wireless network as the target iOS device.

3.4 Manual Acquisition

Manual Acquisition[3] is the process by which the investigator reviews the device's documentation and employs a manual browsing procedure that utilizes the keypad and display features of the device to acquire the needed evidence. This process will not net all of the needed data, especially the deleted data objects. Issues associated with this method include errors in judgment and data modification as well as the incredible amount of time needed to move methodically through all features of the device.

3.5 Chip-off method

Chip-off[3] is a method of acquisition where the investigator physically removes the chip from the device then proceeds to read the device using a secondary device to perform the forensic analysis. This method is very expensive but is able to extract all of the data. In addition, the resulting acquisition can be difficult to interpret and convert. It should be noted that since the drive is always encrypted in the iOS environment, this method has a low degree of success.

3.6 Bit-by-Bit Method

Bit-by-bit method[3] of acquisition is considered the most thorough of all acquisition methods for mobile devices. This method creates a physical bit-by-by copy of the mobile device's data including the deleted files that net in the greatest amount of information. It is considered the method that is most closely related to the traditional methods of evidence acquisition. Unfortunately, in the iOS environment, this method is not possible without the use of jailbreaking.

3.7 Hex-dump analysis

Hex-dump analysis[3] allows for the physical acquisition of mobile device files. This procedure involves connecting the mobile device to an evidence receptacle or removing the SIM card and utilizing a reader then 'dumping' the contents to the receptacle. The evidence retrieved is in a raw format, which requires a data conversion. Access to the deleted files that have not been over-written can be achieved however the nature of the evidence obtained results in inconsistent reporting, is difficult to use, requires custom cables and the source code is often protected by the manufacturer. Additionally, this method is a derivation of the hacker community that may be considered inappropriate in an investigation as is the utilization of the Jailbreaking methodology.

4. ANALYSIS OF IOS LOGICAL DATA

The structure of iOS directory is common for all the iOS devices and is a hub of the entire information. The folder structure is similar to the UNIX layout and the files are stored in text, XML, binary and SQLite database formats. The iOS operating system provides modified, accessed, changed and born times (MACB) that prove to be crucial evidence in any case involving iOS forensic analysis. These MACB times when used with a timeline, generate essential information for an investigation.
Here is some of the data important for investigations [2][10][15]:

- Call logs (/private/var/wireless/Library/CallHistory/call_history.db) - This is an obvious data source when examining a mobile phone. Here we can get a list of people that the suspect has been in contact with, as well as timestamp data. A maximum of 100 calls can be stored in this file and maintains a log of all the missed, incoming and outgoing calls.

- Contacts (/private/var/mobile/Library/AddressBook/AddressBook.sqlitedb) - Contacts mean both phone numbers and e-mail contacts. Today the e-mail contacts could be far more than the usual traditional phone contact. Many phones offer the function to merge these two lists with each other.

- Messages (/private/var/mobile/Library/SMS/sms.db) - Messages here include SMS, MMS and also instant messages which are pretty common nowadays by using a third party message-app on the phone. This enables the user to send messages for free over the 3g/4g network. It contains the text, phone number of messages, the content of the text, etc.

- Internet History (/private/var/mobile/Library/Safari/History.plist) - This information is useful because it lets us see what Internet patterns the person has. We want to see any recent Google searches and visited websites.

- Caches (/private/var/mobile/Library/Caches) - Some of the directories of importance are:

  - appleWebAppCache: Stores the data which is required by the web apps
  - Locationd: It consists of the entire geolocation data of the iOS devices. This file consist of the following files:
    * Consolidated.db: It contains the cell tower and the geolocation data. Location data can be very useful in mapping the person's movements.
    * Clients.plist: Contains the list of applications and services that use the geolocation data along with the information of all the Wi-Fi spots the iOS device has come in contact.

Other useful data:

- Media (/private/var/mobile/Media/PhotoData/Photos.sqlite) - Smartphones are often used as cameras. The cameras on the phones are getting better and better and are pretty handy, thus the interest in extracting these pictures.

- Facebook data - This is a very interesting source of data. Here we can find a whole lot of information about other people in our person's social circle. Facebook is maybe the most widely used social application at this time so this information is very useful. Data means Facebook accounts, friends and maybe check-in locations.

- Deleted files - Deleted files are probably the hardest to extract, since we need a physical copy of the memory on the phone and then carving files out of unallocated space should be performed. But it's nonetheless a good source to try to find information from. Deleted pictures and/or messages could be very valuable evidence.

5. CONCLUSION

Nowadays iOS devices are very popular and therefore during the investigation they are frequently forensically examined. Each upgrade adds to iOS some new features, so technology and versions are being upgraded as well. In this document, we covered basic data structure in iOS devices, files that can be of importance for investigation and methods and tools that are used for extracting as much as available data. Acquisition methods with special attention to logical extraction together with forensic tools are presented. Important artifacts for investigation are photos, videos, contacts, messages, email, GPRS locations, call logs etc. Because of the constant updates of the iOS devices it is important for the forensic examiners to keep up with the changes and to be able to do acquisitions and examination.

6. REFERENCES

[1] How Many iPhones Have Been Sold Worldwide? https://www.lifewire.com/how-many-iphones-have-been-sold-1999500

[2] Mats Engman *Forensic investigations of Apple's iPhone*

[3] Rita M. Barrios, Michael R. Lehrfeld *iOS Mobile Device Forensics: Initial Analysis*

[4] SQLite Databases and Plist Files https://infosecaddicts.com/sqlite-databases-plist-files/

[5] Mona Bader, Ibrahim Baggili *iPhone 3GS Forensics: Logical analysis using Apple iTunes Backup Utility*

[6] iXAM forensics http://www.ixam-forensics.com/

[7] Katana forensics http://katanaforensics.com/

[8] SQLite Database http://www.sqlite.org/about.html

[9] Mattia Epifani, Pasquale Stirparo *iOS Forensics: Where are we now and what are we missing?* https://www.sans.org/summit-archives/file/summit-archive-1492186541.pdf

[10] Tim Proffitt *Forensic Analysis on iOS Devices* https://www.sans.org/reading-room/whitepapers/forensics/forensic-analysis-ios-devices-34092

[11] *Forensic analysis of iPhone backups* https://www.exploit-db.com/docs/english/19767-forensic-analysis-of-ios5-iphone-backups.pdf

[12] Ibrahim Baggili, Shadi Al Awawdeh, Jason Moore *LiFE (Logical iOS Forensics Examiner): An Open Source iOS Backup Forensics Examination Tool*

[13] Cloud Forensics https://www.elcomsoft.com/

[14] IOS Forensics http://resources.infosecinstitute.com/ios-forensics/

[15] IOS Forensic Analysis
http://www.dataforensics.org/ios-forensic-analysis/
[16] Backup and Recovery Overview
https://docs.oracle.com/cd/B14117_01/server.101/b10735/intro.htm
[17] Elcomsoft Phone Breaker
https://www.elcomsoft.com/eppb.html

Expanding the Potential for GPS Evidence Acquisition

Mirjam Skobe, Darja Peternel

Faculty of Computer and Information Science, University of Ljubljana
Slovenia

ms8799@student.uni-lj.si, dp0743@student.uni-lj.si

Abstract. This paper was written based on the paper with the same title [5] which looks into GPS data for investigating purposes. The number of GPS devices and their capabilities have increased immensely during last years which gives the investigators more tools in the forensic procedure and gives criminals more ways to manipulate data in order to mislead the investigators. We will write about GPS network, devices related to GPS, what kind of software is used to gain quality information and possible information collected during an forensic investigation involving GPS receivers.

1 Introduction

Big changes in technology over the past few years have changed the way investigators and offenders look at the crime and how they are using technology as an advantage in every step of the crime/procedure. The knowledge of different devices and their possible usage of GPS has come as a great favor to both of the sides since investigators want to be able to thwart offenders activities whose intentions are to mislead them. The market for Global Positioning Devices (GPS) has grown extremely in the past few years which means that during that time the usage of GPS devices has become more popular between average people. Since in the past the usage of GPS was a privilege on only a few larger devices it has now spread to almost every smaller device and is even more practical to use. We will look at the devices that contain GPS and software that is used on them and what they represent to examiners. While GPS technology has remained almost unchanged the devices that contain GPS have chained so much more. This means that constant changes in GPS devices and their usage of GPS will play an important role in the way criminals and investigators proceed through their steps.

2 Global Positioning System

Global positioning system (GPS) was developed by the United States Department of Defense as a tool for the military. In 1978 the satellite-based system was employed and now consists of 24 satellites orbiting the earth. At first it was used for military operations only but was opened for civilian use from 1980 till 2000. In this time the signal was intentionally degraded because of the military precautions. This was called Selective Availability (SA) . Ground stations support satellite system in a way that they monitor the data sent from satellites and transmit a corrective data back to them. While orbiting the earth satellites sent out two different radio signals, L1 and L2. L1 is set aside for civilian use and transmits data that can be read by civilian receivers to determine location. [5] This signals travel by line of sight, meaning they will pass through clouds, glass and plastic but will not go through most solid objects such as buildings and mountains.[4]

GPS receiver collects and interprets signals from satellites to give the user a fixed position. The precision of the given location depends on the number of satellites the GPS receiver is tracking and some other variables. For the precise evaluation of the data the GPS receiver has to have either a 2-D or 3-D fix on orbiting satellites. When GPS receiver is tracking three satellites it means it has a 2-D fix and when it is tracking four or more

satellites it has a 3-D fix. With 2-D fix the GPS receiver calculates the longitude and latitude of the receiver and also user's movement, where in 3-D fix it adds calculations of the altitude.

2.1 GPS accuracy

Most common factors that can degrade the signal and therefore affect accuracy are:

- **Ionosphere and troposphere delays**: When a GPS signal passes through upper and lower atmosphere the signals get delayed and deflected. The GPS system uses a built-in model that calculates an average amount of delay to partially correct for this type of error.
- **Signal multipath**: Occurs when the GPS signal is reflected off objects before it reaches the receiver. Objects that can create signal time travel delays are tall buildings or large rock surfaces. Multipath errors are particularly common in urban or woody environments and are one of the primary reasons why GPS works poorly or not at all in large buildings, underground, or on narrow city streets that have tall buildings on both sides. [1]
- **Receiver clock errors**: GPS receivers built-in clocks are less stable than the atomic clocks used in satellites. This can produce very small errors that can be eliminated, however, by comparing times of arrival of signals from two satellites (whose transmission times are known exactly).[4]
- **Number of satellites visible**: The more satellites a GPS receiver can "see", the better the accuracy. Buildings, terrain, electronic interference, or sometimes even dense foliage can block signal reception, causing position errors or possibly no position reading at all. GPS units typically will not work indoors, underwater or underground. [4]
- **Satellite geometry/shading**: This refers to the relative position of the satellites at any given time. Ideal satellite geometry exists when the satellites are located at wide angles relative to each other. Poor geometry results when the satellites are located in a line or in a tight grouping.[4]

Wide Area Augmentation System (WAAS) was created to correct some of these errors and also for aircraft navigation. For the purposes of better reported data WAAS consists of a number of ground stations and satellites in order to create a better accuracy for aircraft. Even though WAAS was initially invented for aircrafts it is available for other purposes including civilian use.

Because of the difficulty maintaining a good signal strength within the GPS network LBS was created to help civilians receive signals in foggy cities with high buildings. Location-Based Service (LBS) is a tracking system that uses mobile phone signal. The tracking is done by using GSM cell towers of local mobile phone service providers. Tracking through LBS is less precise when compared to GPS because the device estimates its position in the area of the cell tower. [2]

Most GPS receivers are accurate to within 15 meters on average without SA. As mentioned above this accuracy can be even more accurate with WAAS. Typical WAAS position accuracy on average is 3 meters or even less. There is one more system called DGPS that can give better accuracy as GPS. Differential GPS (DGPS) is a system to provide positional corrections to GPS signals. DGPS uses a fixed, known position to adjust real time GPS signals and it's position accuracy is on average from 3 to 5 meters.[3]

GPS accuracy with or without Selective Availability and WAAS accuracy can be seen on figure 1.

3 GPS devices

Technology growth in the last few years made a big impact on the devices that can host GPS. We can now find a wide range of them on the market. There are only two main categories of GPS devices: built-in and mobile. Where we define built-in devices as permanently placed in a specific area for operation where primary use of the GPS technology is focused on a specific task. These kind of devices are most commonly found in cars,

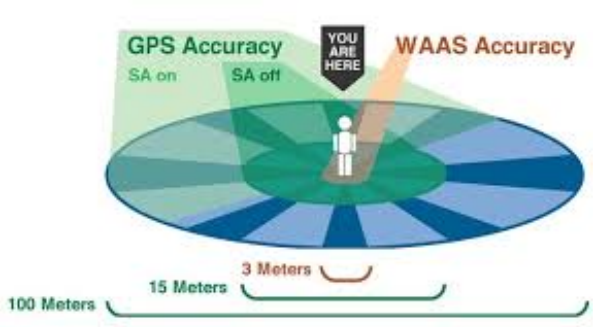


Fig. 1. GPS accuracy depending on the system used. WAAS system can give a 3 meter precision while GPS without SA can only give a 15m accuracy.

aircrafts and marine vessels. They often consist of LCD build into dashboard and are used to output a specific information about the location. They were considered as one of the luxurious options in the past which are now commonly found in standard options in almost every vehicle packages. The advantage for the investigators would be that the devices are built-in which means that they are difficult to remove and it is a high probability that the device was in the car all the time. Which further means that it can contain a lot of information for possible investigation. If device is mobile we are still able to figure out the information about the travel but we can not be sure about the person carrying the device. Mobile devices can also be built-in and are used in automobiles, aircraft and marine vessels. They can be moved around and are small enough to fit into a pocket. The idea was to enhance the effectiveness and capabilities of the device to have only one device for everything instead of more built-in devices each for a specific vehicle.

There has been even bigger growth of devices that are smaller than mobile vehicle devices. Usage of cell phones and smart phones for GPS navigation has become a perfect tool in a device that people have with them almost all the time. The ability to carry a device that is capable of conducting GPS routing is very important because there is no need to carry any other device for GPS navigation. Even though the satellite reception is not the best or at least not as good as found on devices meant only for GPS functions it works pretty well and most importantly it is very handy to use since people always have their cell phones nearby.

But why stop at the size of a cell phone when GPS devices can be even smaller and more practical for sports activities. GPS functionalities can be built into a small unit suitable for wearing around wrists, hung around the neck or put on a bicycle. Most commonly created devices are GPS devices for outdoor hiking/hunting, running, bicycling, golfing and also photographing. These devices normally offer high durability and different functions as topographic maps, electronic compasses and altimeters. Golfing devices can save course data and keep track of the shots and score for the round. A small piece of hardware can be put on the camera in order to embed GPS information inside the picture as latitude and longitude coordinates. This gives the user a chance to identify the location of the picture without having to guess it. There also exist Internet sites that link GPS data with an interactive map and so pictures can be shared with others. Beside GPS devices meant for sports there are certain devices made especially for tracking and logging. This can be done live from distance or analyzed later and can come handy for parents who want to control the movements of their child or for court to track suspect vehicles. Market has also created a device for jamming GPS signals.

4 Software and services

Because of the amount of the GPS devices with new sets of different features and because data and information may not be visible at first sight there are new software packages being released to help investigators examine GPS devices. We will mention two companies that offer such applications:

– **Paraben Corporation**:

  Device Seizure is an all-in-one application that can be used to examine data on multiple handheld devices, like cell phones, PDAs, and GPS units. This application will pull device

settings, maps, waypoints, tracks, and routes from the GPS. In addition to saving this data it can create a *.GPS file that will incorporate all of the point data from the waypoints, tracks, and [5]

Point 2 Point is a software package that can be used in conjunction with the *.GPS file to display all recorded points in Google Earth to gain a map perspective of the locations where the GPS device had been taken.[5]

– **Berla Corporation**:

Blackthorn is specific to GPS analysis and will pull all relevant data pertaining to the data logs that include waypoints, tracks, and routes. The data can then be exported into Microsoft Excel so that it can be easily manipulated and placed into other applications. It also works in conjunction with Microsoft MapPoint and can plot all data points onto a map in a similar fashion as Point 2 Point, in order to visualize the travels of the GPS device. [5]

TomTology is specific to the analysis of GPS units made by TomTom. TomTology essentially does what the other software packages do in that it records all of the pertinent data. These data sets can be viewed in Google Earth just as can be done with Point 2 Point.[5]

The term *geotagging* stands for linking GPS data to digital photos with use of GPS track log and picture timestamps. With the use of applications that support geotagging a user can identify the location of the picture later in time or link image to a map program (as Google Earth). Some applications that allow this: Early Innovations GPS Photo Linker, Francois Schnell's free GPicSync and DeLorme's XMap. As already mentioned there exist GPS devices especially made for logging and tracking of vehicles, valuable objects sometimes even children if parents wish so. LandAirSea and LiveViewGPS offer this kind of services. There are more options where one of them is simply attaching a device to a desired object and tracking it. Since the device is using battery power it can be easily removed and placed onto another object. Devices data tracks can be uploaded on a computer in case of investigation. Other option can be to hardwire a device into a vehicle's power system with the same intention, to track the vehicles position. Life tracking services are also offered by these two companies. Where the location is visible to the user using Internet. Logging into the service allows the user to see the coordinates of the desired object and the usage of special commands performed on the vehicle such as power off the engine or lock the doors. User can be also notified when a tagged object has gone out of the zone that was initially stated as the allowed area. Such as children's backpack or a car.

5 Investigating GPS

At one time gathering GPS data meant an investigator needed only to focus on finding a device whose sole capability was that of a GPS receiver. Nowadays the GPS capabilities have migrated from these specialized devices and have found their way into other types of devices. The reverse is happening as well where dedicated GPS receivers are gaining new capabilities and can handle other types of data as well. This opens up the field of GPS forensics and means that evidence of value may be found in many different devices that are being used for a number of different tasks [5].

This means that more of the crimes that are being committed have the potential of containing GPS information. With all these devices that have recently adopted GPS capabilities, or the GPS units that now offer other types of abilities, the forensic examiner needs to be aware of the potential data that may be located on the device in question. Being able to interpret multiple types of data can help an examiner piece together a scenario that may been unattainable using only a single point of data, or it may create leads that an examiner can follow to turn up more evidence.

Most devices nowadays include additional memory expansion options beyond that of the internal memory installed on the device. These memory modules come in the form of SD (Secure

Digital), or micro-SD, cards that may be inserted, or removed, easily in a unit. The SD card found in a unit may include potential evidence. An SD card found on a GPS unit marketed for backpacking use could very easily be hiding 32GB worth of pirated movies or music.

Even if there are no devices found in the area a search of any computers may lead examiners to believe that such devices do exist and should be sought after. A computer may have software installed that is dedicated to a GPS unit, such as a mapping program, or other applications could reveal a history of uploaded photos that contain geotagging references. Like all investigations, GPS forensics must be approached with a willingness to view a broad picture and the skill to decipher the small details in order to rebuild the occurrence of the crime. Being able to look at a number of physical attributes of the scene can increase the likelihood of revealing GPS units that have been hidden from the investigator.

The forensics analysis of a cell phone and GPS unit are very similar with the addition of so many related features. An investigation on a GPS unit in a vehicle can very likely reveal a wealth of information that includes phone calls made, contact lists, and paired devices. Likewise the cell phone may reveal GPS data and offer up a track created by the individual.

An examiner will want to look at the GPS log to reveal any tracks, routes, or other information that may be stored on the device. The log files with vary in how the data is saved and managed depending on the manufacturer of the device. In most cases an examiner will be able to find 'breadcrumb' trails, which are data points that denote the track created during the travels of the device. The settings used to record tracks will vary upon the device and the settings selected by the user. Understanding the accuracy of the GPS system and the potential errors that produce variances in the reported location of the device can help paint a usable track that plots the whereabouts of the user.

In addition to the plotted tracks there may be saved routes that can be viewed in order to view trips that a user has taken or are planning on taking. Depending on the type of GPS devices routes may consist of turn-by-turn directions along roads or trails, or they can contain data that is a direct route between points. Most devices also allow the creation of custom waypoints and points of interest (POI). Identifying these waypoints or POIs can help an examiner in understanding why the user moved through certain areas and what areas they were planning on visiting in the future. In most cases it is also possible to identify recent location searches, identify the coordinates of a home location, and possibly find information about the owner of the device such as name and phone number.

The functionality of GPS units has drastically changed over the years as they have become integrated into a large number of other electronic devices. These units have become particularly valuable to investigators who are trying to piece together the timeline and movement of individuals during commission of an offense. GPS technology is most assuredly not perfect and an investigator must be aware of the deviations that can occur in data recordings due to errors introduced through hardware or environmental issues.

GPS technology poses a great opportunity for investigators, as it is a source of data that was not so readily available and affordable to the public only a few years ago. The data pulled from GPS devices, which often incorporate more data types than just GPS information, can be accessed and examined typically in the same manner as one would go about investigating data on a computer. One can be assured that GPS technology will remain a central part of many devices in the future and it offers advantages to those on both sides of the law; the game of cat and mouse has escalated with technology and the GPS receiver is commonly involved in the chase.

5.1 TomTom and TomTology software

The TomTom devices are very similar to the other mobile GPS receivers one might encounter in the field. Data is stored via internal flash memory and some units allow an increase to the maximum available memory with the use of an SD card. Hooking the GPS device up to a computer via a USB cable allows a person to view the contents of

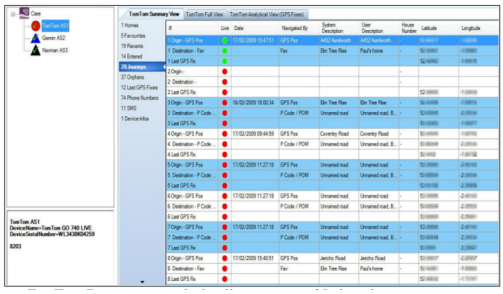


Fig. 2. TomTology Software

the memory on the computer as though it were a removable drive.

TomTom has a number of different GPS models available for purchase with several different features existing on the various models. An examiner who is familiar with the devices or at least able to do a short bit of research on the device collected for evidence, can determine what features are present on the specific model. Most devices contain an owner's information screen where the owner can input personal contact details in case the device becomes lost. Reviewing this screen may reveal name, phone number, and address of an individual.

TomTology 2 simplifies the search of TomTom devices a bit by presenting data in a friendly looking user interface and can automatically mine the device's memory to collect relevant data on the device. Data can be exported and viewed in Google Earth so an investigator can gain a nice visual representation of the path traveled. TomTology is a nice tool for the examiner as a means to collect data in a consistent and usable manner. The only downfall is that it only works on TomTom GPS devices and in the past number of years there have been a number of other companies that have gained a good portion of the market of mobile GPS devices. An investigator may find the software being used sparingly as devices from other manufacturers are being seen more during the collection of evidence.

5.2 DeLorme's Earthmate PN-40

The PN-40 3 by itself is much like any other GPS receiver marketed for the outdoor enthusiast and

Fig. 3. DeLormes Earthmate PN-40

contains the usual data that would be collected from similar devices, such as tracks, routes, waypoints, and points of interest. The device is capable of being loaded with map types other than the typical vector map data this is commonly found on the GPS receiver. This ability alone does not make it more complex to examine, but it becomes a bit more unique against the competition when paired with DeLorme's Topo USA.

Topo USA 4 is an in depth application that allows users to make additions to the data sets and save them to transparent draw layers. This draw layer can then be imported into the PN-40 seamlessly making the data set look as though it is part of the original data set on the GPS receiver. An individual can create a trail that looks like any other, import it to the device, and just looking at the GPS receiver one would assume that the trail was valid. This trail may actually carry a hidden meaning that is only known to the creator of the trail, or select individuals.

The casual observer would not be able to tell that the data was not originally there, but an examiner could look at the data layers to see data has been imported into the device and a draw layer would give good reason to analyze further. If the examiner is suspicious of the draw file for

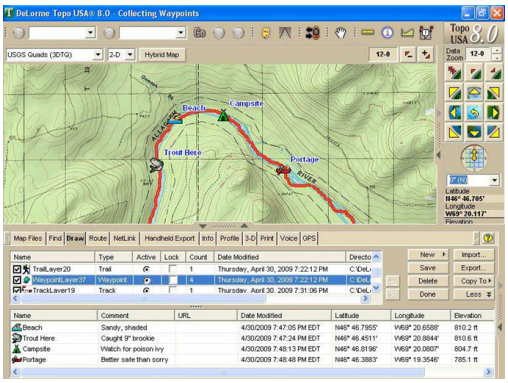


Fig. 4. Topo USA application.

a particular area they may decide to check the other map data layers out for the particular location shown.

6 Conclusion and Future Work

An investigator needs to be familiar with the reporting accuracies being used by the GPS receiver. Modern GPS usage is not impaired by the Selective Availability that was implemented by the United States military, but knowing whether the receiver was utilizing data in a 2-D, 3-D, WAAS or LBS capacity, and the location of the satellites being used for the fix can have a significant impact on the precision of the data log. The accuracy of the data is relevant to the hardware in the GPS receiver as well. Some GPS devices utilize antennas that are more sensitive, or have faster processors, and are capable of getting a satellite lock even inside buildings. It is possible that the GPS receiver was at a specific point in time yet is unreported on the data log because it was unable to lock onto satellites.[5] The functionality of GPS units has drastically changed over the years as they have become integrated into a large number of other electronic devices. These units have become particularly valuable to investigators who are trying to piece together the timeline and movement of individuals during commission of an offense. It is important that the focus of the examination does not rely only on the GPS data collected from devices, but incorporates the findings into the other aspects of the investigation. GPS technology is most assuredly not perfect and an investigator must be aware of the deviations that can occur in data recordings due to errors introduced through hardware or environmental issues. In these paper manly the devices used in the USA were investigated. In future, the investigation of the devices used in Europe could be described, such as Garmin.

References

1. Gps error sources. `https://www.e-education.psu.edu/geog160/node/1924`. Accessed: 2018-06-4.

2. Gps error sources. `http://help.vonino.eu/what-is-gps-and-lbs-tracking/`. Accessed: 2018-06-4.

3. Gps error sources. `https://racelogic.support/01VBOX_Automotive/01General_Information/Knowledge_Base/How_does_DGPS_(Differential_GPS)_work%3F`. Accessed: 2018-06-4.

4. Gps explained. `http://www.gpspassion.com/Hardware/explained.htm`. Accessed: 2018-06-4.

5. Chad Strawn. Expanding the potential for gps evidence acquisition. 2009.

Force Open: Lightweight black box file repair

Karmen Gostiša
Fakulteta za računalništvo in informatiko
Večna pot 113
Ljubljana, Slovenia
kg7155@student.uni-lj.si

Jaka Konda
Fakulteta za računalništvo in informatiko
Večna pot 113
Ljubljana, Slovenia
jk1348@student.uni-lj.si

ABSTRACT
This paper is a summary of a novel approach for automatic repair of corrupted files in study under review [1]. Study presents a lightweight approach that modifies execution of a file viewer, forcing it to open corrupted files. File viewer is referred as a black box that makes this technique file format independent and only requires access to a program binary. According to original authors' results, rate of successfully opened corrupted files in combination with other existing file repair tools was increased. Approach was implemented and evaluated for PNG, JPG and PDF files.

Keywords
File repair, execution hijacking, black box testing, program instrumentation

1. INTRODUCTION
A corrupted file may occur due to a defect or a bug in the software used to process files or at other times a failure of storage media. In many cases these files are not significantly corrupted but only have small corruptions in important parts of the file that prevent file viewer from opening it.

Corrupted file cannot be reconstructed to its original form. However, it is possible to make it usable by reconstructing it to be sufficiently similar to its original. If the file contains most of information contained in the original and a validation program opens it without crashing or error, attempted file repair can be considered successful.

Paper in review is presenting *Force Open*, an approach for automatic file repair that has many advantages over existing approaches to file repair. It does not modify the corrupted file but instead modifies execution of a file viewer, forcing it to open a corrupted file. Force Open is also a black box approach as it is file format independent and only requires access to a program binary.

2. RELATED WORK
Most similar research to the one reviewed here is Docovery [2], a novel document recovery technique based on symbolic execution that makes it possible to fix broken documents without any prior knowledge of the file format. However, Docovery is not a complete black box approach since it requires access to source code of used file viewer. It is also much more expensive approach as it requires symbolic execution.

Similar to the approach presented in this article is technique for automatic input rectification [3]. Its learning phase is also based on collecting information about typical inputs that an application is highly likely to process correctly. Nevertheless, their approach modifies input data by sanitazing it while Force Open approach modifies program execution.

Other research carried out in area of file repair is specific to a single file format and therefore requires in depth knowledge of that specific file format [4] [5].

3. FORCE OPEN
In this section we present some basic concepts to file repair, Force Open approach and evaluation of results.

3.1 Preliminaries
In order to understand the problem we first have to define what is a file corruption. A general definition may be that a file is defined to be corrupted when it contains any form of error. This definition is too broad. For example, a picture may have a single pixel changed and we would not notice anything. This kind of corruption is therefore of little relevance.

In context of this paper, a file is corrupt if it violates at least one of file format specification constraints. A repaired file should contain useful data and it should not lead to false information. Thus a successful file repair must contain as much information as the original file as possible and introduce as little false information as possible.

Most common approaches to file repair try to modify corrupted file so that it meets all constraints. To achieve this, one first has to find all constraints and then solve them in a way that retains as much of original file as possible. Constraint solving can be also highly expensive if file specification introduces complex constraints.

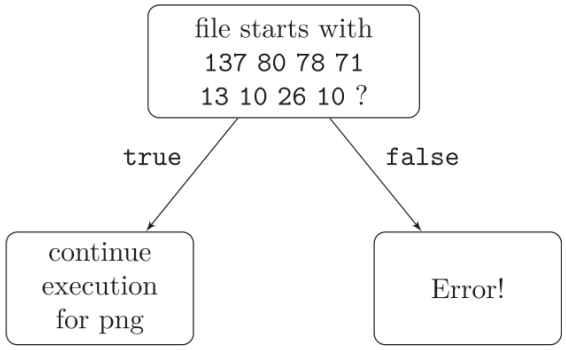


Figure 1: Checking a PNG file signature in a program execution.

Force Open takes advantage of the fact that satisfaction of these constraints are often disclosed in execution of a file viewer. So, if a checksum constraint is satisfied, file viewer will always take same branch. Take for an example a valid PNG file. World Wide Web Consortium issued a PNG specification [6] that declares following file signature constraint:

"The first eight bytes of a PNG datastream always contain the following (decimal) values: `137 80 78 71 13 10 26 10`"

A file that does not meet the constraint above, e.g. a file that starts with `137 80 78` **72** `13 10 26 10` will produce an error. Valid files meet the constraint and the program will always take the true branch (Fig. 1).

Files of same file format share many similar constraints, not only file signature. File viewers make integrity checks to decide if file is valid or corrupted in order to prevent unwanted behaviour such as segmentation faults or security vulnerabilities. A file viewer may thus refuse to open corrupted file even though the file might still be useful if this check is simply skipped. Force Open modifies execution path of the file viewer by forcing execution of branches that would follow an uncorrupted file. To achieve this goal, the tool has to be trained by opening many valid input files with a file viewer and collecting information about these executions. The program then forces execution of the file viewer to open any corrupted file. It does not require access to source code of the instrumented program as it uses binary instrumentation and execution hijacking.

3.2 File repair

Let I be a set of *inputs* and O a set of *outputs*. Both are infinite sets of finite binary sequences, i.e. I may contain binary sequence that represent JPEG file, and O may contain a binary sequence that is sent to output components, such as RGB values for each pixel.

We further define a *specification* $S : I \rightarrow O$ to map inputs to outputs. An input file i is *valid* for a specification S if $S(i)$ is defined, otherwise i is *invalid*. In practice, for any invalid input file program returns an error message. If program fails to detect an invalid file it may also throw a runtime error and crash.

As described in subsection 3.1 a file may become invalid even with minor modifications to it. Consequently file will not be opened in most implementations that follow file format specification.

Most existing approaches to file repair aims to modify invalid file and turn it into a valid with least modifications to original as possible. As described in Section 3.1 this can be hard work due to complex constraints in file format specification.

3.3 Force Open approach

Program works in two phases. The first is called *training phase* and the second *force open phase*. Both are described into details in following sections. Fig. 2 illustrates complete Force Open workflow.

3.3.1 Training phase

In this phase program collects information about branches taken by the file viewer during execution with valid input files. It records branch locations in viewer program (*source*) and locations of instructions that are executed after branching (*dest*). For this purpose we call the `BranchTrace` algorithm (Algorithm 1). It returns a tuple *(source, dest)* for every branch in execution path. Next, we need a training algorithm that gathers information from executions with many different valid input files and combines it in a purposeful way.

Algorithm `SameBranchBehaviour` (Algorithm 2) runs `BranchTrace` for every input file and stores returned branches. It then removes all branches that have same *source* but different *dest*. So the algorithm returns only branches that always have same outcome for all valid files.

3.3.2 Force open phase

The `ForceOpen` algorithm (Algorithm 3) takes a file viewer program, a file and a list of branches as input. It uses execution hijacking to force the behaviour of the file viewer to execute branches leading to successful opening of files. The list of branches is intended to be the result of the `SameBranchBehaviour` algorithm. `ForceOpen` checks all branches in the file viewer and for each one checks if it is contained in the list of branches obtained during the learning phase. If so, the branch is replaced by an unconditional jump to destination (*dest*) for that specific branch. After all branches are verified, the modified program is executed with input file as argument.

3.4 Implementation

Force Open was implemented using the Pin framework [7] that enables writing custom dynamic instrumentation tools, called *pintools*, that work on x86 and x86-64 binaries. By implementing `BranchTrace` and `ForceOpen` as pintools in C++, Force Open approach becomes a black box approach. Pintools also work directly on binaries so the approach does not require source code of the viewing program.

Force Open approach starts with a `BranchTrace` pintool that takes a program and corresponding program arguments as

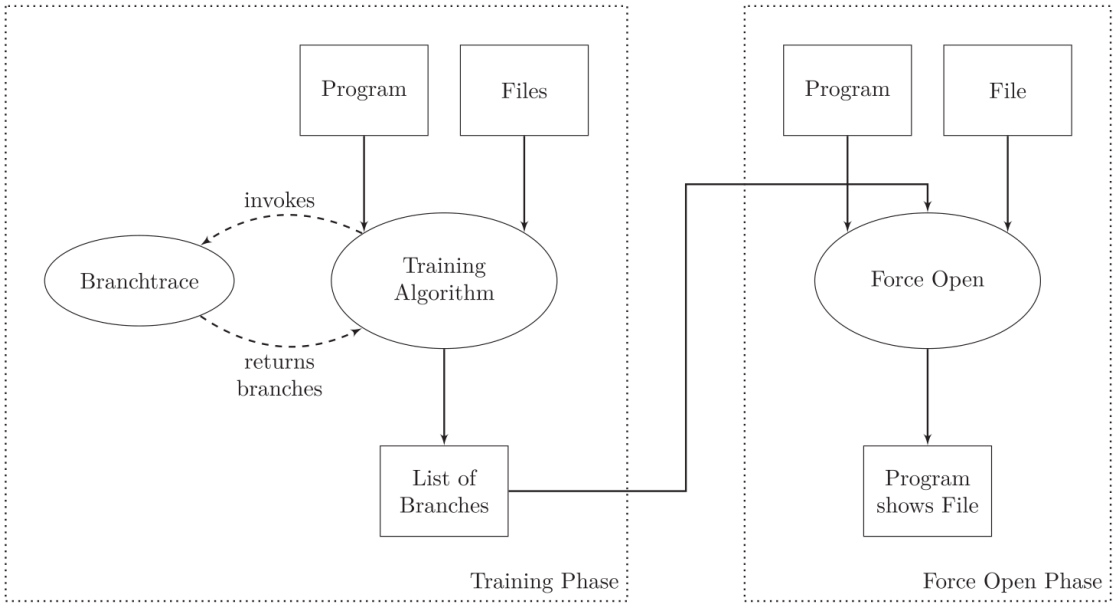


Figure 2: Force Open workflow.

Algorithm 1 Branchtrace

```
function BRANCHTRACE(program, file)
    branches ← empty list
    for all branch in the execution of PROGRAM(file)
do
        source ← location of branch
        dest ← destination taken by branch
        append (source, dest) to branches
    end for
    return branches
end function
```

input. It then inserts a function call after each conditional jump and so it records jump locations and instruction locations that will be executed next. Lastly it executes the program and stores jump locations to a file.

In general, address space of jump instructions changes from one execution to the other. Therefore it is not possible to identify jump instruction only from memory address. Instead, name of module where jump instruction is stored together with offset to the base address of the module, is used.

4. EVALUATION

In this section we present evaluation results of Force Open approach for PNG, JPG and PDF file formats.

4.1 Test conditions and environment

Authors of the original paper used a testing machine with 3.4 GHz Intel Core i7-4770 CPU and 32 GB of DDR3-RAM. It ran a 64-bit Linux distribution. Total time, CPU-time, maximum and average memory usage were measured with the Python script. To increase usability, the process of a file viewer was killed if it took too long to open the file.

Algorithm 2 SameBranchBehaviour

```
function SAMEBRANCHBEHAVIOUR(filelist, program)
    branches ← {}
    for all file ∈ filelist do
        temp_branches ← BRANCHTRACE(program, file)
        branches ← branches ∪ temp_branches
    end for
    for all (source, dest) ∈ branches do
        for all (source, dest2) ∈ branches
        where dest ≠ dest2 do
            branches ← branches \ (source, dest)
            branches ← branches \ (source, dest2)
        end for
    end for
    return branches
end function
```

Algorithm 3 Force Open

```
function FORCEOPEN(program, file, branches)
    for all branch in program do
        source ← location of branch
        if source is a key in branches then
            dest ← branches[source]
            replace branch in program with goto dest
        end if
    end for
    PROGRAM(file)
end function
```

Algorithm 4 Corrupt

```
function CORRUPT(program, file, n)
    repeat
        copy ← file
        position ←u.a.r. {0, size(file) − n}
        copy[position : position + n − 1] ← random
data
    until PROGRAM(copy) fails
    return copy
end function
```

Feh image viewer [8] was used for opening PNGs and JPGs and *pdftotext* Linux command-line utility for PDFs. Results from Force Open for PNGs and JPGs were compared with results of PixRecovery [9], a commercial data recovery program for damaged image files, and for PDFs with results of pdftk [10], a command-line tool for manipulating PDFs.

4.2 Corrupted files generation

According to the definition of file corruption from 3.1 we need files that violate at least one of file format specification's constraints. Algorithm `Corrupt` (Algorithm 4) takes a program, a file and a number n of corrupted bytes as input. It then creates a copy of the file, chooses a random byte-aligned position between 0 and size of the file and then at that position overwrites n bytes with random data. In the end program returns only copy of files that file viewer fails to open.

4.3 PNG

200 valid PNGs files of different sizes were used for training set. Test set consisted of PNG files corrupted with the `Corrupt` algorithm with corruption sizes of 2^k bytes for $k \in 0, 1, 2, 3, 4$.

Training phase needed 35 min to complete and was using 182.39% of the CPU with a maximum memory usage of 167 MB. Results are summarized in Fig. 3. According to expectation, rate of successfully opened files decreases as the number of corrupted bytes increases. It took on average around 15 s of CPU time to open (or fail to open) one image with 1-byte corruption and around 67 s for 16-byte corruptions. Average memory usage was around 250 MB.

Repair rates of Force Open and PixRecovery are comparable. Moreover, number of files that are only repaired by Force Open is larger than the overlap between both tools (Fig. 5). Hereby Force Open can be used to significantly improve existing heuristics.

PNG format supports lossless image compression and consists of an 8-byte file header and a series of chunks. Each chunk is composed of a 4-byte *length* field, a 4-byte *type* field, *data chunk* and a 4-byte *CRC checksum*. Some chunks are critical to decode a PNG file correctly such as IHDR, PLTE, IDAT and IEND. IHDR chunk is the first chunk and it defines attributes such as image dimensions and colour type. PLTE chunk contains colour palette, IDAT chunk contains actual image data, which is the output stream of the compression algorithm and IEND chunk marks end of the file.

Fig. 4 represents types of corruptions for repaired PNG files. We see that certain types of corruptions are handled better than others. As expected, data that is always the same for most files is always repaired.

4.4 JPG

For training set 79 valid JPG files of different sizes were used. Test set consisted of JPG files corrupted with `Corrupt` algorithm with corruption sizes the same as for PNG.

Training phase needed 6.2 min to complete and was using 133.90% of the CPU with a maximum memory usage of 26 MB. Number of repaired files with Force Open and PixRecovery is shown in Fig. 3. Once again success rate decreases as the number of corrupted bytes increases. It took on average around 19 s of CPU time to open a file with a maximum memory usage of approximately 50 MB.

PixRecovery has higher repair rates but Force Open repaired some files that PixRecovery did not (Fig. 5) so the overall repair rate can increase significantly if both tools are used.

4.5 PDF

Training set consisted of 168 text based PDF files (PDFs converted to text with *pdftotext* command-line utility). Again files were corrupted with `Corrupt` algorithm with the same corruption sizes as for PNG and JPG. Output of Force Open was then compared to the output of *pdftotext* on original file using Levenshtein distance as a metric. Levenshtein distance of produced output and output from *pdftotext* is zero if the file is either completely repaired or not at all.

Training phase needed 4 h and 44 min to complete while using 216% of the CPU with a maximum memory usage of 115 MB. Results are shown in Fig. 3. As expected, number of repaired files again decreases as the number of corrupted bytes increases. Time to open (or fail to open) one file is on average 7 s of CPU time with a memory usage of less than 90 MB.

When comparing results of pdftk and Force Open approach, the first is much more successful for larger corruptions and comparable for single byte corruptions (Fig. 3). Overlap of repaired PDFs is small but as Force Open still repairs additional files it can be used as combination to improve overall results.

5. CONCLUSION

After having read and discussed the topic, we are able to conclude that Force Open is a lightweight approach for repair of corrupted files without the need of resource expensive techniques such as symbolic execution. At this expanse the program is not transferable between different version of file viewer binary neither computer architectures and the learning phase needs to be taken for each individually. Another part that should be considered is to sandbox modified application to prevent potential harm if binary file data is executed. Altogether authors showed that Force Open can be used to significantly improve success rate of repaired files in combination with existing tools.

	Corrupt bytes	Number of files	Repaired files									
			Force Open (FO)		Ref. Tool (RT)		FO and RT		Only FO		Only RT	
PNG	1	597	306	(51.26%)	207	(34.67%)	144	(24.12%)	162	(27.14%)	63	(10.55%)
	2	624	305	(48.88%)	174	(27.88%)	119	(19.07%)	186	(29.81%)	55	(8.81%)
	4	612	215	(35.13%)	155	(25.33%)	88	(14.38%)	127	(20.75%)	67	(10.95%)
	8	632	129	(20.41%)	98	(15.51%)	46	(7.28%)	83	(13.13%)	52	(8.23%)
	16	630	54	(8.57%)	58	(9.21%)	25	(3.97%)	29	(4.60%)	33	(5.24%)
	1 – 16	3095	1009	(32.60%)	692	(22.36%)	422	(13.63%)	587	(18.97%)	270	(8.72%)
JPG	1	248	37	(14.92%)	58	(23.39%)	9	(3.63%)	28	(11.29%)	49	(19.76%)
	2	256	30	(11.72%)	45	(17.58%)	14	(5.47%)	16	(6.25%)	31	(12.11%)
	4	249	20	(8.03%)	34	(13.65%)	6	(2.41%)	14	(5.62%)	28	(11.24%)
	8	253	14	(5.53%)	25	(9.88%)	3	(1.19%)	11	(4.35%)	22	(8.70%)
	16	255	6	(2.35%)	44	(17.25%)	1	(0.39%)	5	(1.96%)	43	(16.86%)
	1 – 16	1261	107	(8.49%)	206	(16.34%)	33	(2.62%)	74	(5.87%)	173	(13.72%)
PDF	1	312	30	(9.62%)	33	(10.58%)	3	(0.96%)	27	(8.65%)	30	(9.62%)
	2	320	22	(6.88%)	41	(12.81%)	3	(0.94%)	19	(5.94%)	38	(11.88%)
	4	349	16	(4.58%)	49	(14.04%)	0	(0.00%)	16	(4.58%)	49	(14.04%)
	8	358	9	(2.51%)	48	(13.41%)	2	(0.56%)	7	(1.96%)	46	(12.85%)
	16	384	8	(2.08%)	59	(15.36%)	5	(1.30%)	3	(0.78%)	54	(14.06%)
	1 – 16	1723	85	(4.93%)	230	(13.35%)	13	(0.75)	72	(4.18%)	217	(12.59%)

Figure 3: Results showing number of files repaired with Force Open (FO) and a Reference Tool (RT), which is PixRecovery for PNG and JPG, and pdftk for PDF.

Chunk	Field	# of files	# of repaired files
File header		74	59
IHDR	Chunk type	32	9
IHDR	Chunk length	30	26
IHDR	Chunk data (compression/filter)	25	18
IHDR	Chunk data (other)	106	1
IHDR	CRC	44	33
IDAT	Chunk type	55	20
IDAT	Chunk length	1	0
PLTE	Chunk type	3	0
PLTE	Chunk length	3	0
PLTE	Chunk data	70	33
PLTE	CRC	5	2
Ancillary chunks	Chunk type	72	50
Ancillary chunks	Chunk length	77	49

Figure 4: Types of corruptions for PNG.

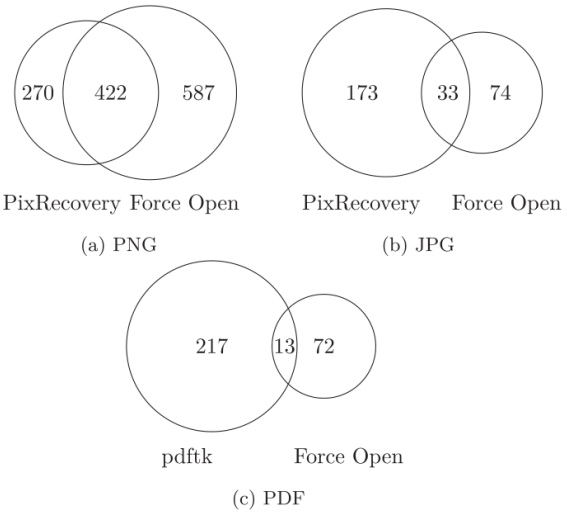


Figure 5: Overlaps of repaired files using Force Open and corresponding reference tool.

6. REFERENCES

[1] K. Wüst, P. Tsankov, S. Radomirović, and M. T. Dashti, "Force open: Lightweight black box file repair," *Digital Investigation*, vol. 20, pp. S75 – S82, 2017. DFRWS 2017 Europe.

[2] T. Kuchta, C. Cadar, M. Castro, and M. Costa, "Docovery: Toward generic automatic document recovery," in *Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering*, ASE '14, (New York, NY, USA), pp. 563–574, ACM, 2014.

[3] F. Long, V. Ganesh, M. Carbin, S. Sidiroglou, and M. Rinard, "Automatic input rectification," in *Proceedings of the 34th International Conference on Software Engineering*, ICSE '12, (Piscataway, NJ, USA), pp. 80–90, IEEE Press, 2012.

[4] R. D. Brown, "Improved recovery and reconstruction of deflated files," *Digital Investigation*, vol. 10, pp. S21 – S29, 2013. The Proceedings of the Thirteenth Annual DFRWS Conference.

[5] H. T. Sencar and N. Memon, "Identification and recovery of jpeg files with missing fragments," vol. 6, 09 2009.

[6] "World Wide Web Consortium (W3C), November 2003. Portable Network Graphics (PNG) Specification, 2nd ed." Available at: http://www.w3.org/TR/2003/REC-PNG-20031110. Accessed: 30 March 2018.

[7] C.-K. Luk, R. Cohn, R. Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, V. J. Reddi, and K. Hazelwood, "Pin: Building customized program analysis tools with dynamic instrumentation," *SIGPLAN Not.*, vol. 40, pp. 190–200, June 2005.

[8] "feh." Available at: https://feh.finalrewind.org/. Accessed: 30 March 2018.

[9] "PixRecovery." Available at: http://www.officerecovery.com/pixrecovery/. Accessed: 30 March 2018.

[10] "pdftk." Available at: https://www.pdflabs.com/tools/pdftk-server/. Accessed: 30 March 2018.

Forenzična analiza deduplicirаnih datotečnih sistemov

Seminarska naloga pri predmetu Računalniška forenzika [*]

Žiga Kern
Univerza v Ljubljani
Fakulteta za računalništvo in informatiko
Večna pot 113
1000 Ljubljana, Slovenija
zk0928@student.uni-lj.si

Tim Oblak
Univerza v Ljubljani
Fakulteta za računalništvo in informatiko
Večna pot 113
1000 Ljubljana, Slovenija
to1702@student.uni-lj.si

Valentin Sojar
Univerza v Ljubljani
Fakulteta za računalništvo in informatiko
Večna pot 113
1000 Ljubljana, Slovenija
vs3598@student.uni-lj.si

POVZETEK

Deduplikacija razdeli datoteke na fragmente, ki so shranjeni v skladišču za drobce (ang. *chunck repository.*) Drobci, ki so sorodni za večje število datotek, so shranjeni le enkrat. S perspektive računalniške forenzike so podatki z naprav, ki uporabljajo deduplikacijo težko obnovljeni, potrebno je posebno znanje, kako tehnologija deluje. Proces deduplikacije spremeni celotno datoteko na organiziran niz fragmentov. Do nedavnega je bila ta tehnologija uporabljena le v podatkovnih središčih, kjer je bila uporabljena za zmanjševanje porabe prostora rezervnih kopij. Zdaj je ta dostopna v odprtokodnih paketih kot je OpenDedup, ali pa kot sistemski dodatek operacijskega sistema. Tak primer je Microsoft z dodatkom v Windows 10 Technical Preview. Orodja, s katerimi se izvajajo preiskave, morajo biti izpopolnjena, da zaznajo, analizirajo in obnovijo vsebino deduplicirаnih datotečnih sistemov. Deduplikacija namreč doda dodaten sloj k dostopu do podatkov. Ta sloj mora biti raziskan, da je zaseg kot nadaljnja analiza izvedena pravilno. V tem članku je predstavljena deduplikacija, ter uporaba v OpenDedup ter na operacijskem sistemu Windows 2012.

Ključne besede

deduplikacija, datotečni sistem, forenzika

1. UVOD

Deduplicirani datotečni sistemi so bili do nedavnega uporabljeni le v produkcijskem okolju, zdaj pa so na voljo tudi navadnim uporabnikom. Digitalna forenzična preiskava ni prisotna le v primerih računalniškega kriminala, vendar v velikih primeru tudi pri preiskavah, ki v osnovni nimajo povezave z računalniki. V veliki večini primerov mora strokovnjak pridobiti neke podatke z datotečnega sistema ali jih obnoviti. Analize, kjer so prisotna podatkovna središča so izvedene z njihovo pomočjo, da so podatki, v primeru deduplikacije, pravilno obravnavani. Večji problem pri preiskavah predstavljajo uporabniški sistemi z deduplikacijo, kot recimo *Windows 10 Technical Preview – 2016*. Brez vednosti o prisotnosti deduplicirаnih enot na pomnilniški enoti, je podatke z nje težko, včasih tudi nemogoče pridobiti.

Uporaba deduplicirаnih datotečnih sistemov da končnemu uporabniku najboljše rezultate v primeru hranjenja prostega prostora. Z uporabo enostavnega deljenja pri Microsoftu [1] je bilo shranjevanje datotek kot so PDF datoteke za 9,96% boljše, pri Office-2007 dokumentih (docx) pa kar 35,82%. Članek vsebuje poglavje Sorodna dela, kjer predstavitev sorodnih del na področju. Temu sledi poglavje o deduplikaciji sami. Temu poglavju sledi poglavje o orodju Opendedup, nato pa še poglavje o Windows 2012. Zadje poglavje je zaključek.

2. SORODNA DELA

Deduplikacija je predstavljena s stališča algoritmov ter njihove učikovitosti pri izdelavah varnostnih kopij [2]. Prav tako je ta tehnika uporabljena v analizi spomina pametnih telefonov. Uporabljena je za zaznavanje podvojenih strani, saj strani v bliskovnem pomnilniku (ang. *Flash*) ob spremembi podatkov niso izbrisane in prepisane, ampak je ustvarjena nova stran, kamor so posodobljeni podatki tudi shranjeni [5]. Število uporabnikov pametnih telefonov je v današnjem času zelo veliko, ker pa vsebujejo pomembne osebne podatke je preiskovanje takih naprav zelo pomembno. Prav tako pa je predstavljena raziskava, ki trdi da je uporaba deduplikacije dobra pri zmanjševanju potrebnega prostora za arhiviranje digitalnih dokazov [3]. Vsaka raziskava ima veliko količino podatkov, katere je potrebno pregledati ter spraviti. Za to bi bilo potrebno imeti veliko prostega prostora, vendar se z uporabo deduplikacije ta količina zmanjša. To pa nič ne spremeni podatkov, ter ne vpliva na njihovo kredibilnost in so lahko prosto predstavljeni na sodišču.

3. DEDUPLIKACIJA

Deduplikacija je proces ki zmanjšuje količino podvojenih podatkov na napravi. Izvaja se lahko na nivoju datotek ali posameznih blokov. Uporabljena je lahko na več različnih področjih. Primer tega je podatkovna deduplikacija, ki je uporabljena v arhivih ter varnostnih kopijah, medtem ko je omrežna uporabljena za zmanjševanje pasovne širine.

Proces deduplikacije je lahko izveden na dva različna na-

[*]Študijsko leto 2017/2018

čina. V prvem je proces izveden neposredno ali takoj (ang. *in-line*), kar pomeni, da so podatki deduplicirani pred shranjevanjem. Ta način uporablja odprtokodni OpenDedup. V drugem primeru, ki se mu reče po procesna deduplikacija (ang. *post process*), pa je proces izveden na podatkih, ki so že shranjeni. Proces pa se izvede na podlagi parametrov, kot so starost, tip in pogostost uporabe določene datoteke. Ta način pa je uporabljen pri Microsoft *Windows Server 2012*.

Proces deduplikacije se izvede na celotni datoteki, z namenom odkritja podvojenih delov, proces je opisan na sliki 1. Vsaka datoteka je razdeljena na fragmente imenovani delci (ang. *chunk*). Za vsake delec je izračunana zgoščevalna vsota (ang. *hash*), pri OpenDedup je uporabljen algoritem SipHash, za Microsoft pa SHA256. Vse nove vsote so shranjene v podatkovno bazo skupaj s kazalcem na mesto v skladišču za delce, kamor je shranjen relativni delec. Vsaka izmed teh vsot predstavlja določen del prvotne datoteke. Postopek rekonstrukcije prvotne datoteke po deduplikaciji se imenuje rehidracija.

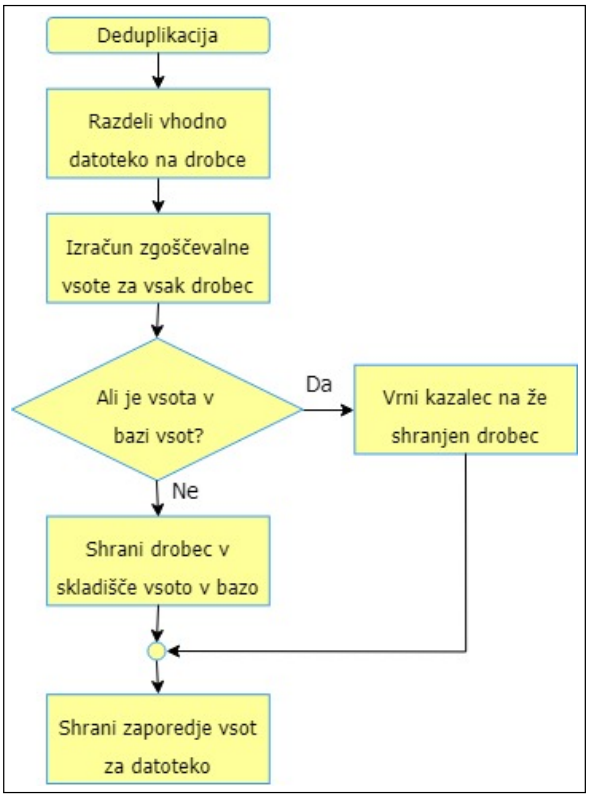
Problem s katerim se srečujejo pri uporabi deduplikaciji je izguba delcev. Vsak delec je shranjen le enkrat. Če se izgubi delec, ki je skupen večjemu številu datotek se datotek, ki so uporabljale ta delec, ne da rekonstruirati. Za reševanje tega problema Microsoft uporablja tehniko večkratnega shranjevanja najpogosteje uporabljenih delcev, OpenDedup pa uporablja SDFS datotečni sistem, ki uporablja več vozlišč, delec pa je shranjen v več njih.

Delci imajo lahko fiksno dolžino, po navadi med 32 in 128 kB, ali variabilno. Pri uporabi fiksne dolžine je računanje vsot enostavnejše, prav tako tudi organizacija. Sprememba datoteke, tudi če zelo majhna, pomeni različno razdelitev na delce, kar pomeni tudi različna vsota. Algoritmi, ki uporabljajo variabilno dolžino temeljijo na določanju delcev s pomočjo prstnih odtisov v tekstu. V primeru da je datoteki dodan le majhen delček, se to pozna le na delcu, ki vsebuje dodan del. Za prepoznavo odtisov je uporabljen Rabin algoritem [6], ki uporablja drseče okno fiksne dolžine bajtov in računa vrednost odtisa z uporabo polinomov. Deduplicirani sistemi za delitev datotek uporabljajo te posebne vzorce prstnih odtisov. Zaradi tega je mogoče obnoviti pogoste delce v podobnih datotekah.

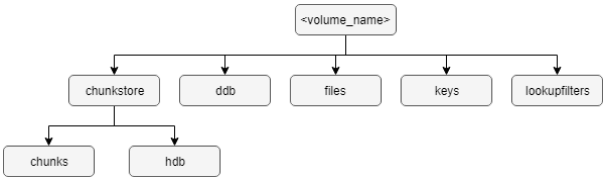
4. OPENDEDUP

Prvi predstavljen datotečni sistem z deduklipacijo je SDFS datotečni sistem, ki je uporabljen v OpenDedup. SDFS je tako imenovani datotečni sistem v uporabniškem programskem prostoru (ang. *Filesystem in Userspace - FUSE*). S SDFS lahko tako ustvarimo virtualni datotečni sistem, ki deluje nad tradicionalnim datotečnim sistemom, kamor se tudi shranjuje spodaj ležeča struktura. Ko pa SDFS priključimo v sistem, se ta obnaša kot tradicionalni datotečni sistem, preko katerega lahko dostopamo do dedupliciranih datotek.

Poleg možnosti shranjevanja podatkov oziroma delcev na lokalnem datotečnem sistemu pa SDFS omogoča tudi hranjenje podatkov na enem izmed ponudnikov objektne hrambe (AWS S3, Glacier, Google Cloud Storage, Azure Blob Storage, ...), kar pa lahko oteži forenzično preiskavo, če nimamo na voljo vseh delcev.



Slika 1: Proces deduplikacije.



Slika 2: Spodaj ležeča struktura datotečnega sistema SDFS.

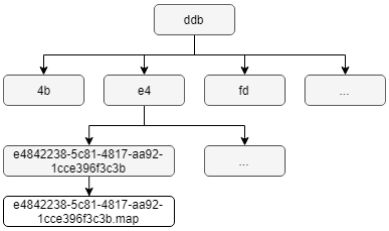
4.1 Struktura

Osnovna struktura sistema, ki je privzeto shranjena v `/opt/sdfs/volumes/<volume_name>/`, je prikazana na sliki 2. Sama struktura je razdeljena na pet map. V mapi `files` se nahaja kopija datotek in map, same datoteke v tej mapi pa vsebujejo metapodatke o originalni datoteki in kazalce, ki so potrebni za rekonstrukcijo datoteke.

Ko si ogledamo vsebino ene izmed teh datotek, ki je prikazana v tabeli 1, lahko ugotovimo iz podpisa datoteke (označeno z modro), da gre za JavaSerialization protokol. Tako lahko sklepamo, da je v datoteki shranjen serializirana java struktura. Iz rozno obarvanega dela, pa lahko razberemo tudi, da gre v tem primeru za ponutdnika OpenDedup, verzijo `3.7.0` (označeno z zeleno) in še bolj specifično za razred `org.opendedup.sdfs.io.MetaDataDedupFile`. Kaj vse točno je zapisano v datoteki poleg velikosti (označene z rumeno), pravic in lastnika, si lahko pogledamo v izvorni kodi, ki je na voljo na Github [4].

Naslov	Šestnajstiški zapis								Tekst
0x000	aced	0005	7372	0027	6f72	672e	6f70	656e	..sr.'org.open
0x010	6465	6475	702e	7364	6673	2e69	6f2e	4d65	dedup.sdfs.io.Me
0x020	7461	4461	7461	4465	6475	7046	696c	65c0	taDataDedupFile.
0x030	2d48	41c2	73c0	350c	0000	7870	7a00	0001	-HA.s.5...xpz...
0x040	07ff	ffff	ffff	ffff	ff00	0000	0000	0159	.........Y
0x050	da00	0001	6353	e5f7	1000	0001	6353	e5f7	....cS.....cS..
0x060	1001	0101	0000	0000	0000	0024	6534	3834	.........$e484
0x070	3232	3338	2d35	6338	312d	3438	3137	2d61	2238-5c81-4817-a
0x080	6139	322d	3163	6365	3339	3666	3363	3362	a92-1cce396f3c3b
0x090	0000	0024	3863	6333	6565	6562	2d66	3836	...$8cc3eeeb-f86
0x0a0	322d	3434	3662	2d61	6336	382d	6634	6465	2-446b-ac68-f4de
0x0b0	3337	3235	6263	3538	0000	0054	0000	0000	3725bc58...T....
0x0c0	0004	0000	0000	0000	0001	6000	0000	0000	.........‘.....
0x0d0	0000	0000	0000	0000	0002	a000	0000	0004	....
0x0e0	6e6f	6e65	ffff	ffff	ffff	ffff	ffff	ffff	none............
0x0f0	ffff	ffff	ffff	ffff	ffff	ffff	ffff	ffff	....
0x100	ffff	ffff	ffff	ffff	ffff	ffff	ffff	ffff	....
0x110	0000	0000	0000	0000	0000	0000	0800	0000	....
0x120	0000	0000	0001	0000	0007	332e	372e	302e	.........3.7.0.
0x130	3000	0000	0000	0000	0000	0081	b400	01ff	0...............
0x140	ffff	ffff	ffff	ff00	78				x

Tabela 1: Vsebina datoteke `dog.jpg` v mapi `files`.

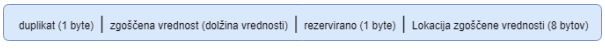


Slika 3: Struktura mape ddb.

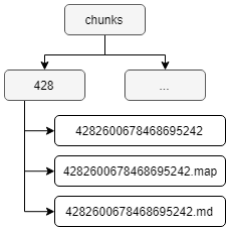
Najbolj pomemben podatek če želimo datoteko ročno rekonstruirati je unikaten identifikator označen z oranžno barvo. S pomočjo tega indentifikatorja lahko v mapi `ddb` najdemo datoteko z istim imenom in končnico `.map`. Ta datoteka vsebuje seznam zgoščenih vrednosti delcev iz katerih so same datoteke sestavljene. Struktura mape je takšna kot je prikazana na sliki 3 in sicer datoteke so razporejene po mapah glede na prvi dve črki indentifikatorja, znotraj teh, pa se nahajajo mape z celotnim unikatnim indentifikatorjem, ki vsebujejo datoteke `.map`.

Vsebina datoteke `.map` je sestavljena iz po vrsti urejenih zaporednih vnosov (Slika 4), ki vsebujejo zgoščeno vrednost delca in indeks v strukturo delcev, s pomočjo katerega lahko najdemo datoteko, ki vsebuje delec.

Za preslikavo med zgoščenimi vrednosti in indeksom s katerim lahko najdemo delce datotek lahko uporabimo tudi podatkovno bazo RocksDB. Ta podatkovna baza vsebuje vse izračunane zgoščene vrednosti delcev na sistemu, podatke pa hrani v mapi `/opt/sdfs/volumes/<volume_name>/chunkstore/hdb`. Ključi v tej podatkovni bazi predstavljajo zgoščene vrednosti delcev, vrednosti pa indekse v strukturo delcev, ki se nahaja v mapi `/opt/sdfs/volumes/<volume_name>/chunkstore/chunks`. Podatkovna baza se primarno uporablja za iskanje duplikatov delcev in sledenje številu referenc na delce. SDFS s pomočjo podatkovne baze tako skrbi, da se delci na datotečni sistem zapišejo le enkrat in da neuporabljene delce briše iz sistema.



Slika 4: Struktura vnosa v `.map` datoteki.



Slika 5: Struktura mape chunks.

Zgoščene vrednosti so v novejših verzijah OpenDedupe (3.5.0 in novejše) privzeto izračunane s algoritmom SipHash 128, pred tem pa so bile izračunane z Murmur3. SDFS pa vključuje tudi druge algoritme, ki jih je moč specificirati v konfiguracijski datoteki, toda se priporoča uporaba hitrejših algoritmov kot SipHash.

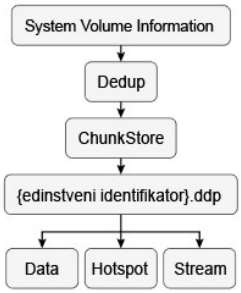
Da s pomočjo indeksa v strukturo delcev najdemo sam delec moramo šestnajstiški zapis (3B6EDAA07AF934CA) tega pretvoriti v desetiškega (4282600678468695242). S pomočjo prvih treh števk desetiškega zapisa lahko potem v strukturi `chunks`, prikazani na sliki 5, najdemo pravo mapo (428) v kateri se nahajajo tri datoteke.

Datoteka `.map` ponovno hrani vrednosti oziroma vnose, ki vsebujejo zgoščeno vrednost delca in vrednost, ki pa predstavlja odmik od začetka datoteke oziroma naslov na katerem se nahajajo podatki delca v datoteki brez končnice. Za rekonstrukcijo datoteke je tako potrebno za vsak delec najti indeks v strukturo delcev in v datotekah poiskati lokacijo in dolžino samega delca ter te delce nato združiti skupaj.

4.2 Forenzična analiza

Za zaseg naprav, ki uporabljajo deduplicirane datotečne sisteme, imamo nekaj možnosti. Najlažja je seveda, da zasežemo celoten sistem ali pa forenzično analizo opravimo kar na prižganem sistemu in datoteke preprosto prenesemo. Naslednja možnost je, da zabeležimo vse podrobnosti in konfiguracijo sistema, kot na primer konfiguracijsko datoteko `/etc/sdfs/<volume_name>-volume-cfg.xml`. S pomočjo konfiguracije lahko potem kasneje sistem repliciramo in dostopamo do podatkov, kot bi dostopali pri originalnem sistemu. Enako lahko seveda storimo, ko imamo za forenzično analizo na voljo le pomnilniški medij, toda moramo v tem primeru sprva identificirati uporabljen datotečni sistem za deduplikacijo in potem poskusiti sistem replicirati.

V primerih ko konfiguracije sistema ne moremo replicirati lahko datoteke poskusimo rekonstruirati ročno ali s pomočjo kakšnega orodja. Žal pa obstoječa orodja za analizo datotečnih sistemov trenutno ne podpirajo dedupliciranih datotečnih sistemov in tako niso v veliko pomoč pri sestavljanju ali izluščevanju datotek iz delcev. Pomagamo si seveda lahko s razlago oziroma analizo sistema kot je predstavljena v prejšnjem poglavju in poskusimo datoteke sestaviti ročno, toda nam delo lahko oteži omogočeno stiskanje ali enkripcija delcev. Pri sestavljanju datotek pa je pomembno, da smo pozorni na verzijo sistema, saj so lahko razlike med verzijami velike. Glede na to, da se datotečni sistemi z deduplikacijo ponavadi uporabljajo za večje količine podatkov, to ni najbolj smiselno. Pomagamo si seveda lahko z avtomatizacijo



Slika 6: Imeniška struktura deduplikacije v Windows 2012.

postopka na podlagi informacij o strukturi podatkov, toda tudi ta proces lahko vzame veliko časa.

Ročna analiza tako pride v poštev samo v skrajnih primerih kot ob na primer korupciji datotečnega sistema. V nasprotnih primerih pa lahko konfiguracijske podatke poskusimo razbrati iz samih datotek, ki jih imamo na voljo. Glavni parametri, ki jih moramo v tem primeru razbrati so velikost in tip delcev, zgoščevalni algoritem, ključ tega in lokacijo delcev ter lokacijo podatkovne baze zgoščenih vrednosti. Velikost in tip delcev lahko ugotovimo relativno preprosto z analizo teh. Tako dokaj hitro ugotovimo ali so fiksne ali variabilne velikosti in v slednjem primeru njihovo največjo in najmanjšo velikost. Ključ zgoščevalnega algoritma najdemo v `.map` datotekah, na algoritem pa lahko sklepamo s pomočjo dolžine vrednosti ali pa poskusimo s privzeto vrednostjo.

5. WINDOWS 2012

Deduplikacija je v operacijskem sistemu *Windows 2012* prisotna le kot ena od možnih razširitev sistema. V nasprotju z OpenDedup ta ni vgrajena direktno v funkcionalnost datotečnega sistema, pač pa deluje na vrhu sistem *NTFS*. Prav tako se postopek ne izvaja med samim shranjevanjem ali ustvarjanjem datotek, ampak se sproži naknadno. Po deduplikaciji je prvotna datoteka z diska izbrisana, ohrani pa se le v obliki razbitih delcev in niza zgoščenih vrednosti, ki predstavljajo preslikavo med delci in njihovo lokacijo v datoteki.

5.1 Struktura imenikov

Proces deduplikacije v operacijskem sistemu *Windows 2012* definira lastno imeniško strukturo, v kateri hrani vse potrebne informacije za delovanje. Struktura je prikazana na sliki 6. Datotečni sistem te informacije hrani v skritem imeniku *System Volume Information*, ki se običajno nahaja v korenskem imeniku pogona z datotečnim sistemom *NTFS*. Tam se hranijo podatki, potrebni za obnovitev sistema, urejanje povezav in bližnic, indeksiranje za hitro iskanje, vse nastavitve, zapisi in stanje procesa deduplikacije pa so zapisani v imeniku *Dedup*. V tem imenuku lahko dostopamo tudi do shrambe posameznih delcev (ang. *Chunkstore*), na katere se datoteke ob deduplikaciji razdelijo. Vsaka takšna shramba je poimenovana z edinstvenim identifikatorjem, vsebuje pa naslednje podstrukture:

Naslov	Vsebina
0x00	C0 00 00 00 A0 0000 00
0x08	00 00 00 00 00 00 03 00
0x10	84 00 00 00 18 00 00 00
0x18	13 00 00 80 7C 00 00 00
0x20	01 02 7C 00 00 00 00 00
0x28	16 8F 09 00 00 00 00 00
0x30	00 00 00 00 00 00 00 00
0x38	E5 90 E4 2E F0 44 9A 4F
0x40	8D 59 D6 D8 A2 B5 65 2C
0x48	40 00 40 00 40 00 00 00
0x50	F5 F4 B2 C1 6E B0 D1 01
0x58	01 00 00 00 00 00 01 00
0x60	00 50 00 00 01 00 00 00
0x68	01 00 00 00 08 05 00 00
0x70	C8 01 00 00 00 00 00 00
0x78	9C FC 06 75 EB 4E D1 0C
0x80	FD 13 F3 14 AA 1D B1 D3
0x88	BC BA 9C 19 E2 EF D5 12
0x90	50 58 CE B1 FB 58 05 00
0x98	C1 AD 45 7A 00 00 00 00

Tabela 2: Točka za vnovično razčlenjevanje v glavni datotečni tabeli.

- Delci dedupliciranih datotek se nahajajo v imeniku `Data`. Ti so lahko shranjeni v obliki, kot so predstavljeni v prvotni datoteki, ali pa so dodatno zgoščeni z brezizgubnim stiskanjem.
- V imeniku `Stream` se nahajajo vsi nizi zgoščenih vrednosti, ki kažejo na shranjene delce. Ti predstavljajo preslikavo med delci in njihovo lokacijo v datoteki.
- Imenik `Hotspot` vsebuje varnostne kopije najbolj uporabnih delcev. Ta zagotavlja redundantnost hranjenih podatkov in minimizira verjetnost izgube.

Strukture, ki so analogne imenikoma `Data` in `Stream`, pogosto najdemo tudi v drugih rešitvah deduplikacije, imenik `Hotspot` pa je posebnost deduplikacije v *Windows 2012*. Če pride do napake in posledične izgube delcev, z vsebino tega imenika zagotovimo, da lahko najpogosteje uporabljene delce obnovimo in tako zmanjšamo velikost izgube.

5.2 Zapis v glavni datotečni tabeli

Običajno se v datotečnem sistemu nahaja datoteka, ki hrani lokacije vseh ostalih datotek, vključno z svojo lokacijo. V primeru *NTFS* je to datoteka *Master File Table*, v nadaljevanju *MFT*. Za vsako deduplicirano datoteko MFT vsebuje vnos, ki hrani informacijo o lokaciji shrambe delcev.

Ti podatki so v *MFT* shranjeni v obliki točke za vnovično razčlenjevanje (ang. *Reparse Point*), v nadaljevanju *RP*. Oblika je prikazana v tabeli 2. Najpogosteje se ta funkcija v operacijskem sistemu *Windows* uporablja za predstavitev relacije med dvema datotekama, kjer ena služi le kot povezava ali bližnjica do druge. V primeru deduplikacije *RP* na odmiku 0x28 (označeno z rožnato) vsebuje s štirimi bajti zapisano informacijo o dolžini prvotne datoteke, na odmiku 0x38 (označeno z oranžno) pa je s 16 bajti zapisan edinstven identifikator shrambe delcev, na katere je bila ta razbita. Na koncu (označeno z oranžno) pa najdemo še kazalec na glavo vsebnika nizov. Vnos v MFT torej predstavlja direktno povezavo med zapisom o obstoju datoteke in lokacijo dedupliciranе različice datoteke.

5.3 Struktura datotek v shrambi delcev

Kot omenjeno v poglavju 5.1, glavna shramba delcev vsebuje dodatne podstrukture. Datoteke v imenikih `Data`, `Stream` in `Hotspot` so shranjene v obliki vsebnikov (ang. *containers*) s

```
<vsebnik niza> := <glava datoteke> <tabela preusmeritev> <niz preslikav>
<niz preslikav> := <preslikava> <niz preslikav> | <preslikava>
<preslikava> := <glava preslikave> <meta podatki> | <zgoščene vrednosti>
<zgoščene vrednosti> := <zgoščena vrednost> <zgoščena vrednost> | <zgoščene vrednosti>
```

Slika 7: Struktura vsebnika niza preslikav.

```
<vsebnik delcev> := <glava datoteke> <tabela preusmeritev> <delci podatkov>
<delci podatkov> := <delec> <delci podatkov> | <delec>
<delec> := <glava delca> <podatki delca>
```

Slika 8: Struktura vsebnika delcev.

Naslov	Vsebina
0x00	43 6B 68 72 01 03 03 01
...	...
0x30	00 00 00 00 00 00 00 00
0x38	9C FC 06 75 EB 4E D1 0C
0x40	FD 13 F3 14 AA 1D B1 D3
0x48	8C BA 9C 19 E2 EF D5 12
0x50	50 58 CE B1 FB 58 0F 27
0x58	EB 47 3C 95 A2 30 E5 A5
0x60	77 51 A6 31 DF FF CB 71
0x68	53 6D 61 70 01 04 04 01
0x70	01 00 00 00 01 00 00 00
0x78	00 50 00 00 01 00 00 00
0x80	2E 5E 01 00 00 00 00 00
0x88	ED DB 30 58 FA 7F 5C 19
0x90	5C 89 FD 23 FE 97 FA 43
0x98	58 B2 99 B4 FF 6B 40 6C
0xA0	0B 8A BE 27 49 BB 28 7A
0xA8	ED A7 00 00 00 00 00 00

Tabela 3: Vsebina vsebnika nizov.

Naslov	Šestnajstiški zapis	Tekst
0x5000	43 6B 68 72 01 03 03 01	Ckhr....
	01 00 00 00 ED A7 00 00	
	01 00 28 00 08 00 00 00	
	08 00 00 00 08 00 00 00	
	02 00 00 00 00 00 00 00	
	ED DB 30 58 FA 7F 5C 19	
	5C 89 FD 23 FE 97 FA 43	
	58 B2 99 B4 FF 6B 40 6C	
	0B 8A BE 27 49 BB 28 7A	
	5D 1A 7C 25 A5 A8 E7 CF	
	32 B8 58 6B BB 92 4C 9D	
	00 00 00 00 50 72 6F 6A	....Proj
	65 63 74 20 47 75 74 65	ect Gute
	6E 62 65 72 67 27 73 20	nberg's
	4C 61 20 44 69 76 69 6E	La Divin
	61 20 43 6F 00 10 00 00	a Co....
	6D 6D 65 64 69 61 20 64	mmedia d
	69 20 44 61 6E 74 65 2C	i Dante,

Tabela 4: Vsebina vsebnika delcev.

končnico `.ccc`. Vsebniki v `Hotspot` predstavljajo le najpogosteje uporabljene vsebnike z imenika `Data`, zato obravnavamo le prvotne vsebnike v `Data`. Tu naredimo pregled nad njihovo strukturo in vsebino:

5.3.1 Stream

Vsebniki v imeniku `Stream` so sestaveljeni s treh delov. Prvi je imenovan *Cthr* in predstavlja glavo datoteke, drugi se imenuje *Rrtl* in vsebuje tabelo preusmeritev, zadnji pa je *Ckht*, v katerem je zapisan celoten niz preslikav. Sintaksa datoteke je prikazana na sliki 7. Če podrobneje pogledamo v binarni zapis niza preslikav, prikazan v tabeli 3, lahko najdemo določene lastnosti strukture tega zapisa:

- Na začetku zapisa se pojavi niz *Chkr* (oznečeno z modro).
- Začenši z odmikom 0x30 od začetnega niza se na vsakih 64 bajtov pojavi nov zapis za posamezno zgoščeno vrednost. Prvi zapis vsebuje le glavo datoteke (značeno z rožnato).
- Vsak nadaljnji zapis vsebuje absolutno pozicijo delca v shrambi (označeno z rumeno) ter zgoščeno vrednost delca (označeno z oranžno).

5.3.2 Data

Tako kot vsebniki v imeniku `Stream`, so vsebniki v `Data` sestavljeni s treh delov; glave datoteke, tabele preusmeritev in množice podatkovnih delcev. Sintaksa datoteke je prikazana na sliki 8. Če se premaknemo na ustrezen naslov, v binarnem zapisu, prikazanem v tabeli 4 najdemo naslednjo strukturo:

- Ponovno se na začetku zapisa pojavi niz *Chkr* (označeno z modro).
- Na odmiku 0x0C (označeno z rožnato) od začetnega niza se pojavi zapis o velikosti delca.
- Za velikostjo ponovno sledi zgoščena vrednost delca (označeno z oranžno), potem pa prvotni podatki (označeno z zeleno), ki so lahko stisnjeni ali pa ne.

5.4 Ostale lastnosti procesa

Delci datotek so lahko stisnjeni, če je tako določeno v nastavitvah postopka deduplikacije. Delci datotek, ki stisnjene podatke že vsebujejo ali pa so šifrirane, so s stiskanja izključeni v vsakem primeru. Algoritem za stiskanje se imenuje LZNT1 in je Microsoftova nadgrajena različica znanega algoritma LZ77 [7].

Proces deduplikacije se proži periodično, čas periode pa določi uporabnik in je zapisan v nastavitvah. Ob izvajanju procesa so upoštevane vse datoteke na disku. V nastavitvah najdemo tudi parameter, ki določa, kakšna je najmanjša dovoljena starost ciljne datoteke. S tem parametrom zagotovimo, da so deduplicirane le datoteke, ki jih hranimo dolgotrajno. Če parameter nastavimo na 0, so datoteke obravnavane ne glede na starost. Prav tako lahko s procesa deduplikacije izvzamemo datoteke z določenimi končnicami.

Velikost zgoščene vrednosti deduplikacije v *Windows 2012* je 256 bitov. V dokumentaciji Microsofta je zapisano, da večina njihovih protokolov uporablja zgoščevalno funkcijo *SHA-1*. Dolžina zgoščene vrednosti delcev se torej ne ujema z običajno uporabljeno zgoščevalno funkcijo, ki jo uporablja Microsoft. Najbolje lahko sklepamo, da je uporabljena zgoščevalna funkcija *SHA-256*.

Če sledimo vsem nizom preslikav v vsebniku, lahko prvotno datoteko povrnemo v prvotno stanje. Če so delci zgoščeni, jih med postopkom sestavljanja najprej pretvorimo v nezgoščeno obliko. V primeru, da deduplicirano datoteko izbrišemo, zapis v MFT skupaj z ustrezno točko za vnovično razčlenjevanje izgine. Delci in nizi zgoščenih vrednosti na disku sicer ostanejo, ob naslednjem čiščenju diska pa so izbrisani.

5.5 Forenzična analiza

V našem primeru je najbolj enostaven način za obnovitev dedupliciranega diska takšen, da ciljni disk priklopimo na napravo z enakim operacijskim sistemom in enako konfiguracijo deduplikacije. Če dedupliciran disk pregledujemo z enim od programov za analizo datotečnih sistemov, program običajno prebere zapise v MFT, prikaže imenike in datoteke, vendar so vse dedupliciranе datoteke prazne. Orodja namreč ne razumejo, kako interpretirati podatke v točki za vnovično razčlenjevanje (RP).

Deduplikacija v *Windows 2012* prvotne datoteke po obravnavi z diska izbriše po običajni poti. To pomeni, da izbriše le reference, podatki na disku pa ostanejo dokler jih ne prepišemo z novimi podatki. To nam omogoči uporabo orodij za obnovitev izbrisanih datotek z diska. Optimizacija diska na stanje dedupliciranih datotek ne vpliva. Ob periodičnem čiščenju diska (ang. *Garbage Collection*) se delci brez ujemajočega vnosa v MFT z diska odstranijo. Pri običajnem čiščenju se izbriše le delež delcev brez vnosa, pri popolnem čiščenju pa se izbrišejo vsi. V vsakem primeru tudi pri periodičnem čiščenju na disku še ostanejo sledi delcev, ki jih lahko obnovimo z ustreznimi orodji.

V primeru poškodovane naprave sta za obnovitev podatkov ključna dva dela s shrambe delcev; vsebnik delcev in vsebnik niza preslikav. Če sledimo nizu preslikav, lahko delce združimo v pravilno zaporedje ter sestavimo prvotno datoteko. Da ločimo med datotekama in razpoznamo njuno strukturo, lahko iščemo oznake in odmike, kot so opisani v poglavju 5.3. Ker niz preslikav vsebuje zgoščene vrednosti vseh delcev, lahko z njimi potrdimo celovitost prisotnih delcev.

6. ZAKLJUčEK

V tej raziskavi smo naredili pregled področja deduplikacije datotečnih sistemov, opisali smo lastnosti dveh dejanskih implementacij, *OpenDedup* in deduplikacijo v operacijskem sistemu *Windows 2012*, poleg tega pa smo analizirali zmožnosti forenzičnega preiskovanja takšnih sistemov. Proizvajalci redko delijo nizkonivojske podrobnosti o implementaciji takšnih sistemov, zato je pomembno, da uporabljene podatkovne strukture in algoritme identificiramo ter omogočimo ustrezno forenzično preiskavo takšnih sistemov.

V idealni situaciji lahko podatke pridobimo tako, da celoten sistem zasežemo, ali pa analizo opravimo na licu mesta, ko je naprava še prižgana. Če je zasežena naprava poškodovana, ali pa zasežemo le podatkovne diske, lahko v primeru obeh rešitev deduplikacije sistem tudi obnovimo. Za to običajno potrebujemo natančno ujemanje nastavitvenih datotek, ujemanje operacijskega sistema in ostalih okoljskih spremenljivk. V skrajnem primeru, če sistema ne moremo podvojiti, se vedno lahko zanesemo na ročno obnovitev podatkov. Ta je sicer dolgotrajna, saj avtomatskih procesov za to še ne poznamo. Trenutna orodja za delo z datotečnimi sistemi takšnega sistema niso zmožna razčleniti, količina podatkov v strežniškem okolju pa je običajno velika. Z analizo procesov deduplikacije tako pridemo korak bližje k oblikovanju orodja, ki bo v prihodnosti takšno zmožnost imel.

7. LITERATURA

[1] A. El-Shimi, R. Kalach, A. Kumar, A. Ottean, J. Li, and S. Sengupta. Primary data deduplication-large scale study and system design. In *USENIX Annual Technical Conference*, volume 2012, pages 285–296, 2012.

[2] Min et al. Efficient deduplication techniques for modern backup operation. *IEEE Transactions on Computersk*, 60(6):824–840, 2011.

[3] S. Neuner, M. Mulazzani, S. Schrittwieser, and E. Weippl. Gradually improving the forensic process. In *Availability, reliability and security (ares), 2015 10th international conference on*, pages 404–410. IEEE, 2015.

[4] OpenDedup. Sdfs. `https://github.com/opendedup/sdfs`, 2018.

[5] Park, Jungheum, et al. Forensic analysis techniques for fragmented flash memory pages in smartphones. *Digital Investigation*, 9(2):109–118, 2012.

[6] M. O. Rabin et al. *Fingerprinting by random polynomials*. Center for Research in Computing Techn., Aiken Computation Laboratory, Univ., 1981.

[7] J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on information theory*, 23(3):337–343, 1977.

Advanced forensic Ext4 inode carving: summary

Jakub Maroušek
Univerza v Ljubljani
Fakulteta za računalništvo in informatiko
1000, Ljubljana
jakub.marousek@gmail.com

Jan Ivanjko
Univerza v Ljubljani
Fakulteta za računalništvo in informatiko
1000, Ljubljana
ji4988@student.uni-lj.si

Jernej Katanec
Univerza v Ljubljani
Fakulteta za računalništvo in informatiko
1000, Ljubljana
jk6071@student.uni-lj.si

ABSTRACT

Many widely-used filesystems (NTFS, FAT, Ext3) are well-researched in terms of digital forensics and data recovery, but this does not apply to Ext4, a successor in the family of Ext filesystems. Due to some new functionalities of Ext4 and compatibility breaking, a novel approach for file carving had to be developed. The advantages of this approach include its ability to restore files even in the case of a corrupted superblock. This article gives a summary of the original paper and its outcomes, also with an introduction to Ext4 filesystem and Ext family included.

1. INTRODUCTION

In the present, Ext4 is a widely-used filesystem, especially on Linux-based operating systems and Android phones [7]. The filesystem, released in 2008, has its origins in Ext2 and Ext3 filesystems. While many of their principles and structures were preserved (it is possible to upgrade from Ext2 or Ext3 in-place by setting new attributes), the changes are not just evolutionary. The way of storing and locating data blocks has changed, especially with the introduction of *extents*. From the forensical point of view, it was necessary to come up with new processes of recovering data.

This article provides a summary of the paper [7]. But since the method is not easily understandable without wider information about Ext4 way of working, we include here a description of the filesystem. Furthermore, a description of Ext2 and Ext3 must be also attached, as Ext4 is closely tied to them and succeeds their functions.

Our work is organized in the following way: Section 1 is this introduction; Section 2 provides the description of the filesystems; Section 3 is the summary of the paper; in Section 4 results of the evaluation are described; and Section 5 is the conclusion.

2. INTRODUCTION TO EXT FILESYSTEM FAMILY

The Ext4 filesystem is the next generation in the family of Ext filesystems, used typically in Linux environments. In the very beginning, the Linux kernel used Minix filesystem, due to the fact that Linux itself originated from Minix. This filesystem, however, contained serious limitations, the maximum file name size and the maximum filesystem size as the most severe examples. A group of kernel developers, with Theodore Ts'o being the most prominent one, introduced a couple of new filesystems [6].

The first new filesystem, Ext ("extended filesystem", released in 1992) removed the Minix limitations, but some of the previous problems remained – for example, free blocks were monitored over a linked list, which harmed performance. As a quick response, Ext2 ("second extended filesystem", 1993) was released. While the code of Ext2 originated in that of Ext, it brought many notable improvements.

2.1 The Ext2 filesystem

The filesystem supports standard Unix file types: regular files, directories, device-special files and links. The maximum size of a file is 2 GB, the maximum size of the whole filesystem is 4 TB. As the other Unix filesystems, it also uses inodes for storing file metadata. An *inode* is a structure containing a description of a file: file type, access rights, owners, timestamps, size, pointers to data blocks.

A directory is only a special kind of file, with a list of files and their corresponding inodes. A link can be either hard or symbolic. A hard link is basically a file pointing to the same inode as the original file, not distinguishable from the previous file. To remove a hard-linked file, all the references to the file must be removed. A symbolic link contains a text specifying a path to the file. Unlike the symlink, creating a hard link has limitations: it is possible to hard-link only a file from the same partition (due to the nature of the link) and hard-linking directories is not allowed (to prevent infinite subdirectory loops). A device-special file is only a pointer to the device driver.

While these features are common for all Unix filesystems, now we shall list some abilities specifical for Ext2. Using the file attributes, the kernel behavior of creating new files in a directory can be modified, specifically the new file user id and group id. The logical block sizes can be specified when the filesystem is created – larger block sizes lead to

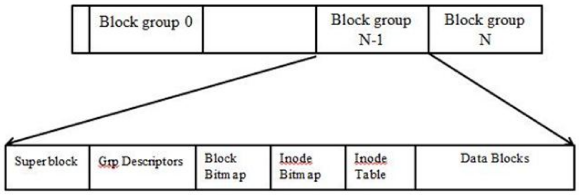


Figure 1: The structure of an Ext2 filesystem [2]

efficient behavior, but also waste more space.

The symbolic links of Ext2 can use more efficient way of storing the path inside the inode of the symlink. The overall status of the filesystem is saved in the superblock. During the mounting of the filesystem, if it is detected that it was not unmounted properly, the filesystem is automatically checked. After a certain number of mounts or when a certain time period has passed, the filesystem checking is enforced.

It is also possible to create *immutable* files, which either cannot be written into or deleted, or they can be opened for writing, but the text is appended to the end of the file.

The filesystem structure is as follows. There is a boot sector in the beginning of the space, which is designated for bootloaders code. The rest of the drive is split into *block groups*. Each block groups contains inodes (in an inode table) and data blocks. An inode bitmap and a block bitmap marks which inode/block is used and which is not.

The superblock and group descriptors contain information which are crucial for the filesystem functioning, and thus they are put in the beginning of each group as a backup. The superblock contains the basic information about the structures, like logical block sizes, the number of blocks and inodes, positions of data blocks and so on. An array of group descriptors contains pointers to the inode table and the block table for every present group.

As the main optimization, Ext2 performs block readahead, that means, reads more continguous blocks at once. Block groups try to keep together inodes and blocks of a file, which reduces disk seeks. Also when a file is being written, Ext2 preallocates up to 8 blocks.

2.2 The Ext3 filesystem

After the initial period of bugs elimination, the Ext2 filesystem became a safe, stable choice for a Linux-based operating system. One of the main drawbacks of using Ext2 is the lack of jounrnaling, which slows down the filesystem check and makes files prone to corruption.

First suggestions for the journaling of ext filesystem family were put down by Stephen C. Tweedie [5]. He identifies several aspects of filesystem reliability and puts forward *preservation* ("stable" data on the disk should never be damaged), *predictability* (the failure modes of the filesystem should be predictable and deterministic) and *atomicity* (every operation is either fully performed or fully undone).

Figure 2: The structure of Ext3 journal [12]

The Ext2 provided only the preservation aspect, whilst the failure states (could occur because of an unexpected reboot, for example) were not properly defined and might have prevented the operations from fully performing.

The Tweedie's article [5] compares approaches of reaching the desired aspects. The easiest way is to wait for a disk write to complete before the next one is submitted, which breaks the performance. Another idea lies in preserving multiple write buffers, ordering them accordingly and write these separately. However, one can easily get into a situation when the buffer dependence is cyclic (moving a file from directory A to directory B and, in the same time, moving another file from B to A). While there exists a mechanism to solve it, all the presented approaches require the disk to be completely rescanned for errors in case of a system failure. (Recall that filesystem check slowness was one of main objections against Ext2.)

Instead, the author proposes to use *journaling*. To ensure operations to be atomic, a batch of data is written on the disk, but is not effective until a special *commit* block does not conclude the operation. Thus, a failure during performing the write can result only in two possibilities: either the commit block has been written on the disk, which means the transaction is complete, or it has not been written, in which case the transaction as a whole is undone. More practically: there is a dedicated place on the disk, called a *journal*, working as a circular buffer. A transaction starts with a description block, continues with all the blocks that the transaction is modifying, and end with a commit block. Only after all these blocks have been successfully written to the journal, the "real" blocks modification is done. If a failure of the system occurs, the recovery program goes through the journal and performs all transactions which contain the commit block. If the block is not present, the transaction is discarded.

The Ext3 filesystem, released in 2001, offers three different modes of journaling: [12]

- *Journal* – All data and matadata are logged. This approach minimizes the chance of losing data, but has the laregst performance penalty.
- *Ordered* – Only metadata are logged, but file data are flushed on the disk before the change of metadata occurs, keeping the journal synchronized with data writes. This is the default mode.
- *Write back* – Only metadata are logged, file data are flushed whenever possible. This mode is the fastest, but also the most dangerous.

Notice that journaling cannot be turned off. Compared to Ext2, this is the only different feature. Thanks to this, it is

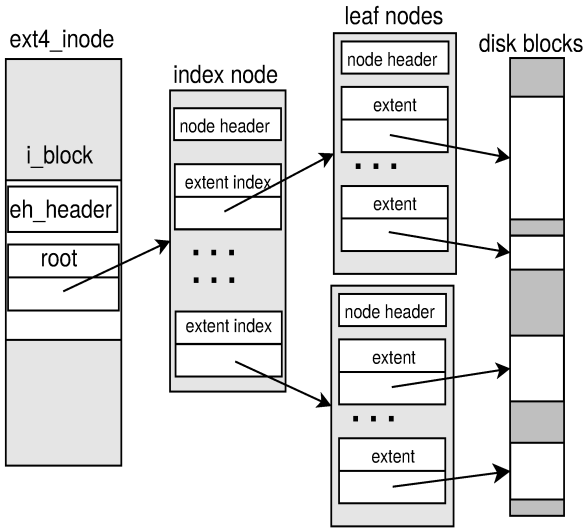


Figure 3: Extent tree layout [10]

possible to convert an Ext2 partition to Ext3 with spawning a single command. For the period of Ext3 development, a principle of both backward and forward compatibility with Ext2 was closely followed by its authors. Therefore, an Ext3 partition can be mounted with Ext2 driver and works the same with the exception of journaling. Turning off a journal causes an Ext3 partition to become Ext2 and vice versa.

2.3 The Ext4 filesystem

Although Ext3 was a step forward, it still lacked some state-of-the-art features, which could be found in other Unix filesystems. Between 2003 and 2006, numerous patches for the Ext3 were proposed to be added to the Linux kernel, but they were not accepted because the Linux community was afraid of stability of their mostly-used-filesystem at that time. As a result, Theodore Ts'o introducted plans for creating a successor of Ext3, which would partially break the forward compatibility, but still allowed an easy update from the older versions [10]. In 2008, the code was marked as stable.

A large amount work was done in terms of scalability. The maximum size of the filesystem was raised to 1 EB, the maximum size of a file is raised to 2 TB. The maximum number of files is increased too.

A compatibility-breaking change of file block locations is brought up. Instead of storing a list of data blocks (which is advantageous for small files but unpractical for large ones), *extents* are introduced. An extent is a range of data blocks, where all blocks in the range belong to the file. Four extents can be directly stored in the inode of the file. For large and fragmented files, a tree of pointers is maintained, with extents being in the leaves.

The transactions in the journal are checksummed in the commit block, making two consequences: the recovery is more reliable as the corrupted transaction blocks are detected and the transaction cancelled; also now the commit block can be written in the same time as the other blocks, which significantly improves the performance.

Contrary to Ext4, persistent pre-allocation of file size is possible. The checking of the filesystem is 2 to 20 times faster, as the unallocated inodes are marked and are skipped during the rest of the procedure. Instead of allocating one block at a time, the allocation is *delayed* in Ext4, meaning it is postponed in time and performed when the file is flushed. There is also a possibility of online defragmentation. Since the size of an inode is larger, nanosecond resolution timestamps are supported and the problem of year 2038 is deferred for 272 years.

Similarly as in the case of Ext2 and Ext3, migrating to Ext4 is just a matter of turning on the new features. The legacy system of individual block mapping is preserved, but all new files will use extents. Downgrading from Ext4 to Ext3 by turning off the features is straightforward when extents are not used – otherwise, temporary copy of the files is needed.

2.4 Comparison of Ext4 and other Unix filesystems

It is worth noting that Theodore Ts'o considered Ext4 not to be a major advance and still using old technology. In the light of his words, we found potentially interesting to list other popular Unix filesystems and compare their functions.

Btrfs [13] is a filesystem brought to life by Oracle. The development began in 2007 and the software was proclaimed to be stable in 2014. The name comes from the B-tree as this is the main structure used within the filesystem. Notable features of Btrfs are copy-on-write (a copied and unmodified file references to the "old data"), snapshots, CRC checksums of data, RAID or configurable compression. In order to prolong the lifespan, SSD drivers are handled specifically. In-place conversion from Ext2/3/4 and ReiserFS is present.

ReiserFS was created by company Namesys with Hans Reiser as the chief developer. When the first version was released in 2001, it brought attention because of features which had been unavailable, like journaling, online filesystem resizing, packing files in a single disk block or B-tree usage. Although ReiserFS was popular in the past, it was overtaken by other filesystems since then. The main reason may be the slow pace of development, especially after Hans Reiser has been imprisoned and convicted of murder.

ZFS [3] was developed by Oracle, but after the end of support the project was handled to the open-source community [1]. It supports journaling, read-only snaphots of the files state, RAID or checksums of files. On Ubuntu, ZFS has been chosen as the default filesystem for containers.

3. FILE CARVING IN EXT4

File carving, or sometimes just "carving" is the process of extracting data collection from a larger data set. Digital investigation often include data carving techniques when unallocated filesystem space is analysed to extract files. These files are carved from the unallocated space using file type-specific header and footer values[11].

File carving is not the same as file recovery. File recovery techniques rely on the filesystem information that remains after the deletion of a file to recover those files. On the other hand file carving techniques are used to restore as much data or data fragments as possible, when the filesystem is corrupted or deleted. Carving or raw data recovery process does not rely on the filesystem structures. It searches block by block for data matching the specific file type header and footer values[4].

In digital forensics carving is especially useful in criminal cases, becouse it can recover evidence. As long as data on a disk is not overwritten or wiped, it can be restored using file carving techniques. Sometimes even data from formatted drives can be restored if the conditions are right. The most common general techniques to carve files are[9]:

- **File Structure Based carving** This technique uses identifier string, header, footer and size information to assume internal layout of a file. Header is a unique identifier, its value identifies the type of a file. Its existence means we can identify the beginning of a file, while the existence of a footer shows the tail of a file. The blocks between the header and the footer represent the targeted file. In some cases file format has no footer, therefore a maximum file size is used in the carving program.
- **Content based carving** Carving based on content structure (MBOX, HTML, XML) or linguistic analysis of the file's content. A semantic carver might conclude that some blocks of German in a middle of a long HTML file written in English is a fragment left from a previous allocated file, and not from the English HTML file. Similarly for other content characteristics like, character count, information entropy, white and black listing of data.

Carving can be classified as basic and advanced, with basic it is assumed that the beginning of file is not overwritten, the file is not fragmented and it is not compressed. Advanced carving relies on internal file's structure and occurs even to fragmented files, where fragments can be non-nsequential, out of order or missing.

Since basic carving does not consider the file's content the attention has shifted to advanced carving methods. Header and footer are not enough to carve files because the file's content is not checked nor is sector within header/footer examined. Deeper knowledge of internal file's structure results in less false positives.

Authors of the paper present an advanced method that uses pattern-based file carving. It searches for metadata structures of inodes to recover their content data. This approach avoids reading the superblock and group descriptor table since its goal is to recover files from reformatted or corrupted Ext4 filesystems. The presented method can be divided into five phases [8]:

1. **Initialization**

   The point of initialization is to gather Ext4 parameters, which can be estimated from the filesystem size.

   The relevant parameters are filesystem size, block size, inode size, inode ratio, flex group size, number of blocks per block group, number of blocks and block groups in the filesystem and number of inodes per block group.

2. **Inode carving**

   Not every 128 byte permutation forms a valid inode. For an inode to be correct, some of the values must be in certain relations. This fact is used while searching for potential inode candidates in a byte wise manner. The most significant 4 bits in the 2 byte structure of inode indicate its file type.

   Search patterns can also be based on timestamps, such as time interval or its inner consistency. Creation date, modification date, deletion date timestamp consistency can be verified if the following conditions are true:

   - Modification date < creation date
   - Deletion date = 0 and (deletion date > modification date and deletion date > creation date
   - Modification, creation time and delete time must be valid

   Extent header field must contain the statisticaly defined magic number 0xf30a. Other inode attributes can be used for search patterns. All found addresses of potential inodes are sorted by file type (regular files and directories) and used for recovery.

3. **Directory tree**

   This phase tries to identify potential directory inodes. Directory entries are searched linearly where directories not strating with '.' or '..' are discarded. The file name and inode number are saved along with its parent inode number, therefore a logical tree forming the complete file path can be deducted.

   The module must map physical inode addresses to inode numbers. The beggining of the inode table can be computed by equation:

$$s = (bg_a * n_{BG} + o_s + o_i + o_r) * b$$

   The mapping of an address to its inode number is computed by:

$$o_s = min\{1024, 1 + \left\lceil \frac{d * n_g + 1024}{b} \right\rceil\}$$

$$o_s = 2 * n_f$$

$$X = \begin{cases} 0, & \text{if } b = 1024 \\ 1, & \text{otherwise} \end{cases} \quad (1)$$

$$bg_a = \left\lfloor \frac{a}{b * n_{BG}} \right\rfloor$$

$$f(a) = (\frac{a - s}{i} + n_{i,BG} * bg_a + 1)$$

   where:

- b is the block size
- bg_a is the block group of the address a
- n_{BG} is the number of inodes per block group
- o_s combined size of the superblock, the group descriptor table and growth space
- o_i=2*n_f, the offset where n_f is flex group size
- o_r First 1024 bytes of the filesystem are reserved independently of the block size. This offset is in case the block size is 1024 and the reserved space takes whole block, shifting all addresses by one block.
- i size of inode

The mapping from physical addresses to inode numbers can be done using these formulas. By using information from directory trees, file names can be associated to inodes. In case of a irreparable directory, its children's file paths can not be reconstructed, therefore their content is saved in files named after their inode address. All recovered files are named after their inode address and saved into a flat hierarchy.

4. **Regular files**

   Having reconstructed file path or not, the content of a regular file is now recovered. Ext4 filesystem stores file content scattered across the volume in a way managed by the extents. Since the filesystem journal can contain copies of the existing inodes, duplicates with different physical addresses can be found. In order for two inodes to be the same, the file size and the extent structures are compared, due to the lack of inode numbers.

5. **Files without content**

   This is another optional phase and builds on the results of the directory tree and content data phase. The content of files found in the directory tree phase but not in the inode carving phase cannot be recovered, but their filepath can. Therefore files are created empty with their original filename and file path.

4. EVALUATION OF THE METHOD

In the previous sections, an approach for reconstructing inodes was introduced. It is based on search patterns of different inode attributes. All inodes that match the patterns are considered potential inodes. Using their extent tree, the file content can be reconstructed. In this section, we present an evaluation of the quality of each search pattern and also the completeness and correctness of presented tool. It was firstly introduced in a 2017 paper by A. Dewald and S. Seufert [7].

4.1 Dataset

They have built a dataset with different hard disk images to evaluate their approach and Sleuthkit implementation of the tool. They have used different filesystem sizes to cover different configurations that can occur when formatting disks of different sizes with default values.

Further, they have built test cases where they deleted specific files or changed filesystem parameters that might influence the success of the approach. They distinguished two cases for images where they have deleted files: files that have been deleted directly and files that have been deleted by moving to the trash first. They also made a case where they deleted all files and copied new ones on the filesystem to check if the former files can be recovered. Further, they compared Ext4 filesystems with enabled and disabled journal and deleted files. To create a more realistic example image, they created one that contains an entire Ubuntu Linux installation with various files that have been moved, deleted and modified and also containing symbolic links, device files and other non-regular file types to cover a broad spectrum. At the end, they created images where they overformatted the existing Ext4 with NTFS or again Ext4. For each of those cases, they compared standard and quick formatting. The dataset contained mostly small images.

Pattern	Hits	t-miss	a-miss	Select
access rights	151 k	1.02 M	99.9 M	0.6%
time interval	72.3 k	207 k	238 k	0.003%
time consist	209 k	5.71 M	1.42 G	8.89%
link count	201 k	29.8 M	7.59 G	47.6%
extent flag	166 k	13.4 M	3.28 G	20.6%
extent header	166 k	515 k	53.2 k	0.004%
file type	151 k	4.51 M	905 M	5.7%

Table 1: Search pattern selectivity

4.2 Search patterns and selectivity

In the paper they checked how well the different patterns perform. To test this, they chose the image with installed Ubuntu and various cases. They used the Sleuthkit tool fsstat to provide a ground truth about the number of reserved inodes. Considering the first 10 inodes to be reserved, there remained 201.269 files of which they checked how many are regular files and directories.

Each pattern was tested on each physical address, accepting or declining an address as a potential inode. Accepted addresses that lie within an area of an inode table at a valid offset and can be reconstructed successfully are listed as hits in the table. This is the total number of files in the filesystem. Table misses are structures that might be a valid inode, but reside outside an inode table. Those could be false positives, but could also be copies of inodes for example in the filesystem journal. Similarly, address misses are potential inode addresses that do not lie at a 128 byte inode boundary.

Misses do not necessarily mean false positives, so they are not rejected at the first place. The number of accepted addresses for the pattern is the sum of t-misses and a-misses, while the selectivity of each pattern is the number compared to the total number of possible addresses.

Table 1 shows the number of identified potential inodes for each pattern. 'k' stands for a thousand, 'M' for a million and 'G' for a billion, t-miss are table misses and a-miss are address misses. The selectivity is defined as hits compared to all misses.

In the following sections, we discuss the results for each pattern in detail.

Owner	Group	All
r - -	r - -	r - -
rw -	r - -	r - -
rw -	rw -	r - -
rwx	r - x	r - x
rwx	rwx	r - x
rwx	- - -	- - -
rw -	- - -	- - -
rwx	r - x	- - -
rwx	rwx	- - -

Table 2: Access rights patterns chosen for the experiment

4.2.1 Access rights

While there is no illegal combination for access rights, a search pattern can be defined to search for files with specific rights or to cover the most common access rights combinations. In the configuration file of the proposed tool, the pattern can be adjusted. For the experiment, they choose the most common combinations of access rights in Linux systems, which are shown in Table 2. This pattern accepted only 142 855 valid inodes out of 201 269. However, this contained almost all regular files and folders.

4.2.2 Timestamps

The timestamp pattern consists of two parts: firstly, they validated the inner consistency of the different timestamps, and in the second part, the investigator might configure a relevant timeframe for his case. For this experiment they choose the timeframe from 2015-01-01 00:00:00 GMT to 2016-01-01 00:00:00 GMT, in which the system was set up and all actions have been performed. Amongst the potential inodes have been 7 140 false positives, so that the number of hits was reduced to 65 170.

With respect only to the inner consistency, they obtained 209 481 hits, 5 708 816 t-misses and 1 416 653 929 a-misses. 8 213 of the hits have been identified as false positives, however, all other valid 201 264 have been found.

4.2.3 Number of hard links

In the paper, they have checked whether the internal number of hard link counter of a potential inode is larger then 0. All 201 269 potential inodes were accepted by this, beside special inodes 7 and 8, which are counted as false positives.

4.2.4 Extent flag and header

Researchers evaluated the pattern for the extent flag and extent headers separately. The selectivity of the second header is very high, because of the 2 byte magic number. Its pattern is the only pattern that produces far more table misses that address misses. This pattern also matches various content data blocks. However, all regular files and folder have been accepted by this pattern, along with 8 074 false positives. Also, inodes of deleted files are accepted by this pattern too.

4.2.5 File type

The authors of the paper were restricted only to regular files and folders, so they adjusted the file type pattern accordingly. Besides 8 068 false positives, this pattern matches all 142 919 inodes they searched for.

4.2.6 Pattern combination

In this experiment, 3 most restrictive patterns were the extent headers, timestamp intervals and access rights. Last two include some semantic filtering, so researchers did not include them in the next experiments to verify correctness of the tool.

4.2.7 Completeness and correctness

To evaluate completeness and correctness of the tool, both operation modes of the tool were tested and compared. Correct recovery was checked with MD5 and SHA256 hashes. In the contentdata mode, the files need to be reconstructed with the correct values, while in the metadata mode also the file name and path have to be correct, in order to ensure correct recovery.

The authors of the research evaluated two images of different categories. First, they evaluated the tool on the real case images to verify correctness and completeness by comparing the results of the tool to the known ground truth. Second, they wanted to evaluate whether they can recover files from filesystem that have been overformatted.

4.3 Real world cases

Researchers presented some real world cases. In their first case, the metadata mode reconstructed empty files with the correct file names and paths for all non-regular files and all files and folders that have been placed correctly. On the other hand, in the contentdata mode, folders were not reconstructed, but all the placed files had been recovered correctly.

They created one image by sending all files on the image to the trash. The trash contained all the original files, as well as metadata files that documented the original paths and times for deletion, which all had been recovered by the tool.

In the next step, they emptied that trash. In metadata mode, the tool only recovered the empty root and trash folder. But in the contentdata mode, all files were reconstructed correctly, just without original file names.

They went further and they deleted all files from images with sending them to trash and emptying it. After that they copied new files to the resulting new images. The metadata mode recovered all newly written files correctly from all images. The contentdata mode was also able to recover one file form the original filesystem, whose file content had not been overwritten by new files. Its inode resided in the old journal.

4.4 Overformatted filesystems

In this section, the evaluation, if it is possible to reconstruct files from Ext4 that have been overformatted in different ways, is given. The researchers evaluated this on small image, which provided an upper boundary of what files can be reconstructed.

They overformatted the original file with Ext4 and NTFS

Image name and mode	All inodes	Regularfiles	Folders
small_fastExt4.img	**128**	**121**	**7**
accepted		171	77
metadata mode		118	5
contentdata mode		119	0
small_fastdiffExt4.img	**128**	**121**	**7**
accepted		228	90
metadata mode		0	0
contentdata mode		125	0
small_fastNTFS.img	**128**	**121**	**7**
accepted		348	94
metadata mode		121	7
contentdata mode		124	0

Table 3: Number of found inodes after selection of the overformatted dataset

filesystems in full formatting mode. In those cases, the blocks had been zeroed and they were not able to recover any files. Ext4 as well as NTFS provide an option for so called fast formatting, where no blocks are zeroed while formatting and only newly used blocks are overwritten.

The image `small_fastExt4.img` had been created by using the same default parameters when overformatting the volume as used for the original formatting. The tool recovered all but 3 of the original files and all but 2 folders in the metadata mode, while in the contentdata mode it additionally recovered the journal from inode 8 as a file.

In `small_fastdiffExt4.img` non-default parameters had been used for formatting. None of the original inode table resided in areas that had not been overwritten and the metadata mode was not able to recover files. The journal had not been overwritten, so that in the content data mode, the tool was able to recover 125 files, of which 52 were totally unmodified, but the others had been partly overwritten.

In the image `small_fastNTFS.img` , the structures that had been created for the NTFS filesystem did not overwrite the area of the original inode tables and both operation modes were able to recover all the original files, of which only some had modified content and some blocks had been used by the new filesystem.

Table 3 lists the number of found inodes and files from the fast overformatted filesystems and compared results from both modes of the tool.

4.5 Runtime performance

Both implemented methods show similar speed, while the contentdata mode is slightly faster. The runtime is linear with respect to the image size. Both approaches take about 30s per GB image size.

5. CONCLUSIONS

In this article, we have introduced the readers to the structure and functioning of Ext2, Ext3 and Ext4 filesystems. For Ext4, we have presented a novel approach of data recovery, which is able to restore data even in case of a corrupted superblock [7]. The solution was proved to be successful in many situations and was adapted for Sleuthkit framework.

6. REFERENCES

[1] Openzfs announcement. `http://www.open-zfs.org/wiki/Announcement`. Accessed: 2018-05-11.

[2] The second extended file system (ext2). `http://www.linux-tutorial.info/modules.php?name=MContent&pageid=272`. Accessed: 2018-05-10.

[3] What is zfs? `https://docs.oracle.com/cd/E23823_01/html/819-5461/zfsover-2.html`. Accessed: 2018-05-11.

[4] C. Beek. Introduction to file carving. *White paper. McAfee*, 2011.

[5] S. C. Tweedie. Journaling the linux ext2fs filesystem. 05 1999.

[6] R. Card, T. Ts, and S. Tweedie. Design and implementation of the second extended filesystem. 01 1990.

[7] A. Dewald and S. Seufert. Afeic: Advanced forensic ext4 inode carving. 2017.

[8] A. Dewald and S. Seufert. Afeic: Advanced forensic ext4 inode carving. *Digital Investigation*, 20:S83–S91, 2017.

[9] A. Hadi et al. Reviewing and evaluating existing file carving techniques for jpeg files. In *Cybersecurity and Cyberforensics Conference (CCC), 2016*, pages 55–59. IEEE, 2016.

[10] A. Mathur, M. Cao, S. Bhattacharya, A. Dilger, A. Tomas, and L. Vivier. The new ext4 filesystem: Current status and future plans. 01 2007.

[11] A. Merola. Data carving concepts. *Sans Institute*, 4, 2008.

[12] G. Narváez. Taking advantage of ext3 journaling file system in a forensic investigation. 2007.

[13] O. Rodeh, J. Bacik, and C. Mason. Btrfs: The linux b-tree filesystem. 9, 08 2013.

Linux memory forensics: Dissecting the user space process heap

Reviewing and verifying a cutting-edge solution for Linux memory analysis

Jan Makovecki
University of Ljubljana
Faculty of Computer and Information Science
Večna pot 113
Ljubljana, Slovenia
jm5619@student.uni-lj.si

Sašo Stanovnik
University of Ljubljana
Faculty of Computer and Information Science
Večna pot 113
Ljubljana, Slovenia
ss3055@student.uni-lj.si

ABSTRACT

This work is an overview of a paper [4] on a new method of performing memory forensics on the heap of Linux user space programs. We present common structures found in the `glibc` implementation of the process heap and their general organization. We summarize how the knowledge of these structures can help locate program data in memory, confirm none of it was overlooked and separate it from heap's metadata and data belonging to other programs. We also give a general overview of Rekall, the framework used in exploring the memory, test out the new method and verify the reproducibility of the source paper's results on both old and new versions of analysed software.

Categories and Subject Descriptors

D.4.2 [**Operating Systems**]: Storage Management—*Virtual memory*

General Terms

Design, Experimentation

Keywords

Linux, memory, forensics, user-space, heap, Rekall

1. INTRODUCTION

The process heap of user-space applications has traditionally been looked upon as a singular area of memory in which a running program's data is stored, without much thought of its inner workings. When a certain piece of information was to be located within the heap, a text search was usually ran over the entire heap, looking for patterns and information that seemed to be structured in a correct way. The authors of the source paper [4] took a different approach to the problem of locating and extracting data of a running program.

1.1 Related work

The fact that the operating system and user software, even the ones that are security-focused, preserve passwords in memory, has long been a known fact [6]. Tests have shown that in the memory regions of software such as display managers, mail clients, the `su` executable end even the TrueCrypt encryption utility, plaintext passwords can be found. This issue is not only present for live machine analysis, but also for machines that have been shut off for short or long periods of time.

Even after a computer has been shut off, data may remain in physical memory. This is true even if there is no active electrical signal to the memory modules and even if the modules themselves are removed from the machine in question. At room temperatures, memory may not be as volatile as we expect for a few minutes, and that time is extended if we lower the temperature of the memory modules in order to preserve data for analysis until reconnecting them to a power source.

A longer-term memory retention "vulnerability" is the fact that the swap file or partition is present by default on most distributions. As potentially confidential data is swapped out, that data remains on disk forever (until overwritten) and may be analysed freely, without even the need of performing a memory dump.

An analyst may learn how a program's passwords are stored by performing tests, and then use such signatures of locations and surrounding memory layout and characteristics to reliably obtain data from a live system. The source article [4] expands on that idea by analysing high-level heap structures and using that information to have a better view into the program's memory layout. This can help more efficiently search for data that does not have a distinct signature.

On the topic of memory acquisition, there are two ways to capture a program's or a system's memory for analysis, both of which supported by Rekall, as we will learn later. The first method is creating a memory dump, similar to a disk image, before analysis, and then operating on that. The second, more ad hoc, is running the analysis directly on the live system. Both approaches have their benefits and an investigator needs to be aware of them and their drawbacks.

A drawback of live memory analysis is that forensics tools may be vulnerable to malicious code and that they may have an effect on the memory they are analysing [3]. When running live analysis tools, up to 25 % of terminated processes that have initially still been present in memory, albeit unallocated, are overwritten by new data. The effect is exacerbated with systems with less memory. Even when running a memory dump program on a live system, some data may be overwritten, but frequently less than half than that of a full forensics investigation software.

1.2 General Approach

Instead of treating the heap as one, large area of memory, the researchers took a look at its implementation in recent versions of the GNU C library, which is the heap implementation used in majority of applications that run on different distributions of the Linux operating system. Because the implementation is open-source, they were able to study the inner structure of the heap and leverage that knowledge to look for information in a more sophisticated and effective manner. By identifying specific structures inside the heap they were able to separate the structures of the heap from program data, detect that data was missing in their searches and expand them to locate it, and draw connections between different parts of data that were used together in the given program.We will take a look at the heap dissection process in section 3, but for now let us focus on the structure of the heap.

1.3 The Glibc Process Heap

The process heap can be implemented in many different ways and different operating systems, as well as some programs inside the same one, usually use different implementations. The one that we will examine is the version implemented by the GNU C Library (glibc) version 2.23 [4].

At the highest level the heap is divided into multiple **arenas**, that are represented by *malloc_state* structs. Each arena governs one or multiple virtual memory areas, described by *vm_area_struct* structs. Below them we find *heap_info* structs, that can be found in all arenas, except the main one. At the lowest level the data is stored inside **chunks** that are stored together in "bins" and described by *malloc_chunk* structs.

1.3.1 Arenas

We already mentioned a "main" arena, which is the only one necessarily allocated for each program and kept inside the continuous, main heap area [5]. It belongs to the main thread of the program, while all other arenas belong to one of the additional threads or are shared between multiple threads. Each non-main arena has virtual memory areas assigned to it, and inside those the actual chunks with data are stored.

All the arenas are linked together in a circular linked list via their *malloc_state* structs. These are located right after the first *heap_info* struct of a thread's arena, before its actual chunks begin. Aside from the pointer to the next one, each *malloc_state* struct contains a pointer to its "top chunk", which is the last chunk of the arena and marks its ending, a pointer to its freed chunks (as opposed to allocated ones, more on that in section 1.3.4), a counter for the number of threads attached to the arena, where 0 represents a free arena, as well as some additional data about the arena. These structures and the structs that describe them are summarized in table 1 and visualized in figure 1. Note that in the figure we display arenas as a concept, they are actually described in *malloc_state* structs that are not shown in the image.

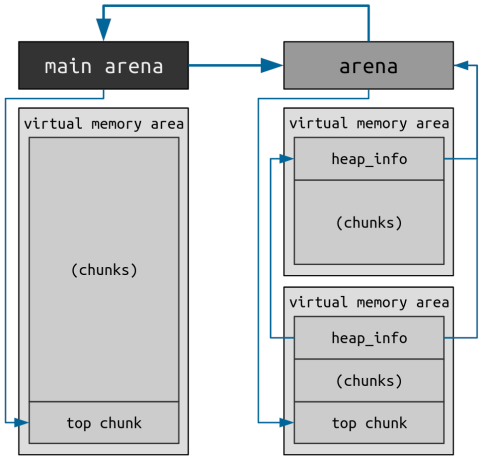


Figure 1: High-level overview of the glibc process heap.

Table 1: Overview of memory structures

Component	Struct
arena	malloc_state
virtual memory area	vm_area_struct
given VMA's heap information	heap_info
chunk description	malloc_chunk

1.3.2 Virtual Memory Areas

These are the areas of memory that are assigned to programs by the Linux kernel via a *mmap()* system call. They are continuous, do not overlap with one another and are described in *vm_area_struct* structs. When a virtual memory area belongs to a non-main arena of a program heap it contains at least one *heap_info* struct at its beginning.

1.3.3 Heap info structs

The main purpose of *heap_info* structs is to hold the size of heap parts contained in their memory regions. They each also contain a link to their respective arena, as well as a pointer to the previous *heap_info* struct of their arena (possibly located in a different memory region), thus forming a linked list.

1.3.4 Bins and chunks

Chunks are the smallest elements of the heap and are generally divided into allocated and freed ones. Allocated chuks hold data that is currently in use by the program, while the freed chunks hold data that was previously in use, but was freed. The struct that holds data describing a chunk

is called *malloc_chunk* and holds a number of fields, not all of which are in use in all varieties of chunks. These fields are *prev_size, size, fd, bk, fd_nextsize* and *bk_nextsize*. The struct is located at the beginning of each chunk and its *size* field is always in use. The *prev_size* field is set to 0 if the previous chunk doesn't exist, its size if the previous chunk is a freed chunk, or simply gets overwritten with data if the previous chunk is allocated. In allocated chunks, every other field (aside from *size*) gets overwritten with data as well.

Aside from always being in use, the size field is also special because its lowest 3 bits function as flags. If the previous chunk is allocated, the lowest bit (*PREV_INUSE*) is set. If the current chunk is MMAPPED (further explained in section 1.3.5), the second lowest bit (*IS_MMAPPED*) is set and if it is located outside of main arena the third lowest bit (*NON_MAIN_ARENA*) is set.

Freed chunks use more of *malloc_chunk*'s fields, but still not necessarily all of them. Which fields are used depends on what type of **bin** the chunk is located in. There are three types of bins, which mainly differ in size: **fastbins** hold chunks smaller than 80 bytes, **small bins** hold chunks below 512 bytes and **large bins** hold anything larger. Fastbins only use *fd* field, that points to the next freed chunk. Small bins extend that by also using *bk* field, that points to the previous freed chunk. Large bins use all the fields, including *fd_nextsize*, which points to the next chunk that is bigger than the current one, and *bk_nextsize*, which similarly references the previous bigger chunk. A comparison of used fields for chunks in different types of bins is shown in table 2.

Table 2: Fields used in different types of bins

Field	Fastbins	Small bins	Large bins
size	yes	yes	yes
fd	yes	yes	yes
bk	no	yes	yes
fd_nextsize	no	no	yes
bk_nextsize	no	no	yes

1.3.5 MMAPPED chunks

MMAPED chunks are an exception in memory structures. They are typically created when a program attempts to allocate a large area of memory that exceeds a given threshold, say 128 KiB. MMAPED chunks exist simply as chunks, in separate, exclusive areas of memory. These areas are created by the operating system after being sent a (*mmap*) system call, the same way that parts of heaps that belong to arenas are, but contain no structures describing them as arenas and no *heap_info* structs. These chunks therefore exist outside of arenas and *heap_info* structs and aren't connected to existing chunks and structures via any pointers either – they are simply independent, allocated chunks. Since the *mmap()* call allocates memory in pages, the chunks allocated directly by it can only be in sizes that are multiples of one page, which is typically 4 KiB. They are also aligned to addresses that are divisible by page size (the size of a page on a given Linux system can be checked via command `getconf PAGE_SIZE`). Size and alignment work similarly for non-MMAPPED chunks as well, as they use

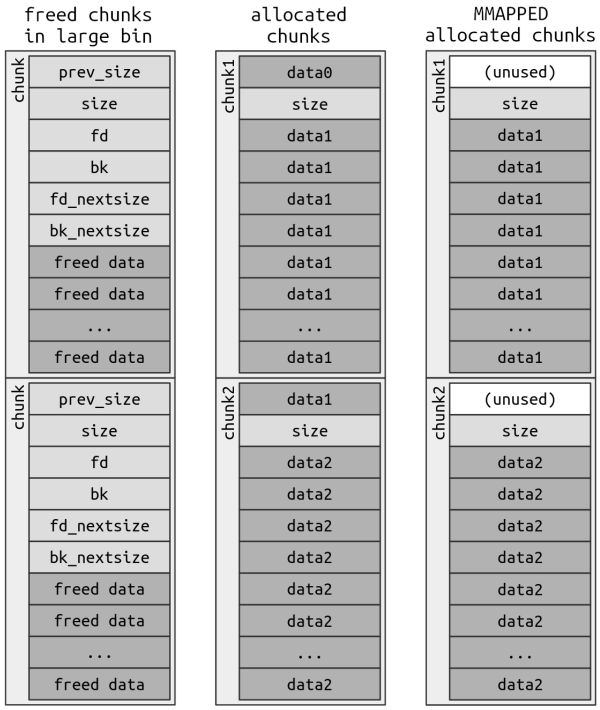


Figure 2: Fields are used differently in different types of chunks.

multiples of 8 or 16 depending on processor architecture (x86 or amd64).

Like allocating, freeing MMAPPED chunks works differently from freeing regular ones as well. As soon as a MMAPPED chunk is freed, its memory is returned to the operating system. A side effect of this is that there isn't a real guarantee that any MMAPPED chunk is followed by another one, so the *prev_size* field, that usually stores the last bit of data from the previous allocated chunk, remains empty.

A comparison between freed, regular allocated and MMAPED chunks can be seen in figure 2.

2. THE REKALL MEMORY FORENSICS FRAMEWORK

Rekall [1] is a kernel memory forensics framework forked from the Volatility [2] project. It supports on-line and offline acquisition and processing of memory data and mapped files and works on Windows, mac OS (former OS X) and Linux. The fact that there is no need to collect a separate memory dump and execute the analysis on a separate system makes it easier to quickly triage a running system with information obtained through calling system APIs.

The project is delivered as a Python library, making it possible to include the framework in custom scripts and integrating it into other applications through its exposed APIs. However, because it is licensed under the GPL 2.0 license, it

cannot be included in products with an incompatible license.

There may be a historical stigma against GUI tools for power users, especially on Linux, but Rekall aims to provide a useful, optional graphical interface similar to IPython Notebook. It allows users to perform the complete acquisition, analysis and processing procedure within the interface to simplify the process and enhance discoverability. Another feature is the ability to export a non-interactive version of the document used for reporting.

Rekall's main mode of operation is different from memory analysis programs created before it. Where previous techniques relied on scanning the memory for interesting strings, Rekall uses the host OS debugging information directly, which makes searches more robust and allows for a stronger API for plugin writers and more advanced detection procedures that transcend simple signature matches. However, it is only capable of searching the kernel memory constructs, not user-space memory, which [4] aims to extend.

2.1 Analysing a live system

We used Ubuntu 16.04.1 as our base system, as the current latest version of Ubuntu, 18.04, does not yet have a profile created for it by the Rekall project, which makes out-of-the-box analysis harder. Figure 3 shows an example output for the Rekall `bash` plugin on a fresh system.

The procedure to install and run Rekall on a fresh system is clearly shown. `bash` history entries are sorted by time; however the times are slightly inaccurate. Notice that the initial machine setup steps used to install the VMware Tools package all have the same timestamp and that they are ordered in reverse.

After the initial virtual machine setup, the user installed Rekall into a Python 3 virtual environment and switched to the `root` user. We see two separate bash processes running and analysed in two separate tasks. The second task is run as the superuser in order to perform live memory acquisition with `rekall -live Memory`.

Inside the interactive Rekall interface, the only command run was `plugins.bash`, which triggers the analysis and displays the results. The whole process takes less than a second. The `zsh` analysis plugin developed in the source paper[4] is already part of the Rekall core and does not need to be installed separately. It is invoked by running `plugins.zsh`.

When first invoking the plugin, the execution fails. This may be because of a completely untested change when trying to achieve compatibility with Python 3 and it manifests itself as needing to convert a byte-string into a bytes object manually in the source code of the zsh plugin. Specifically, the *b* in this line needs to be added: `command = command[:command.index(b"\x00")]`.

Running the command with `plugins.zsh(pids=[26784])`, where 26784 is the process ID of the currently running `zsh` process successfully finds all executed commands, as claimed in the source article. These tests were performed without any debug symbols for glibc installed, therefore the fuzzier searching method used in the article is also functional.

Table 3: Detection of values between KeePassX versions

method	v0.4.3	v2.0.2
plugin title	yes	no
plugin url	yes	no
plugin username	yes	no
plugin password	no	no
yarascan title	yes	no
yarascan url	no	no
yarascan username	no	no
yarascan password	no	no

2.2 Security improvements in analysed software

As the version of KeePassX that was analysed in the source paper is quite old (available on e.g. Ubuntu 14.04), we decided to see whether the analysis technique and plugins still work after updates to the analysed software.

The `zsh` analysis worked on the most recent version of `zsh` available on Ubuntu 16.04 (5.1.1), and that version is older than the one used in the source article [4]. All analysis worked equally well on both versions.

However, the version of KeePassX used in the article (0.4.3) is a legacy version, which does not support the newer database format introduced by KeePass. We were unable to replicate any results on the latest version of KeePassX available - 2.0.2. We therefore executed an in-depth analysis of what the latest Rekall is able to find with regards to KeePassX versions.

Table 3 shows what values are detected in the two versions of KeePassX. We used the plugin alongside *yarascan*, another Rekall plugins that allows searching the memory of a process for a specific string. All analysis was done with a fresh database, adding a new entry with a title, URL, username and password, the saving the database and then once again opening the entry window and revealing the password.

We first note that the results for v0.4.3 under the source article's plugin match what was described in the source article. Next, we note that no information can be discovered by either the `keepassx` or the `yarascan` plugins. This means that the way that the information is stored inside the program has changed to be less discoverable in the newer version, however this does not imply that the newer version is more secure—an analysis of the same scale as creating a newer version of the plugins would need to be performed to claim that. Additionally, *yarascan* is only able to find the entry's title in the older version.

3. DISSECTION METHODS

Using Rekall 2 the authors of [4] created a python class, **HeapAnalysis** that allows a user to use the previously described knowledge of heap's components and layout to analyze it in detail. The class is a basis for a number of other classes that function as general–purpose heap analysis plu-

```
[1] Live(/proc/kcore) 21:57:13> plugins.bash
------------------------------> plugins.bash()
         timestamp         command
  ------------------------ -------
-----------------
Task: bash (6942)
-----------------
  2018-05-12 19:51:42Z     cd vmware-tools-distrib/
  2018-05-12 19:51:42Z     sudo ./vmware-install.pl
  2018-05-12 19:51:42Z     sudo apt update
  2018-05-12 19:51:42Z     sudo apt install build-essential python3 python3-dev dkms
  2018-05-12 19:51:42Z     cd Desktop/
  2018-05-12 19:51:46Z     mkdir rekall
  2018-05-12 19:51:48Z     cd rekall/
  2018-05-12 19:51:58Z     sudo apt install python3-venv
  2018-05-12 19:52:12Z     python3 -m venv .venv
  2018-05-12 19:52:16Z     source .venv/bin/activate
  2018-05-12 19:52:57Z     pip install --upgrade pip setuptools wheel
  2018-05-12 19:53:09Z     pip install rekall
  2018-05-12 19:55:54Z     sudo su
------------------
Task: bash (18112)
------------------
  2018-05-12 19:56:01Z     source .venv/bin/activate
  2018-05-12 19:56:05Z     rekall --live Memory
Out<21:57:14> Plugin: bash (BashHistory)
```

Figure 3: An example output of the Rekall bash plugin.

gins with specific functions:

- **heapinfo** counts the number of arenas and chunks, as well as displays their sizes
- **heapdump** extracts all allocated and freed chunks that it discovers and saves them to disk
- **heapsearch** allows all discovered chunks to be searched for a string or pointer or via regex
- **heaprefs** searches the data part of chunks in order to find pointers to any other chunks

3.1 Locating the main arena

The class *HeapAnalysis* contains a number of functions for specific tasks that are used in writing plugins. One of the main ones, used in initialization of the class, is *get_main_arena*, which locates and returns information about the main arena of the program whose memory we're analyzing. Finding the main arena has a number of uses, as it is the first in the linked list of arenas, allowing us to locate the others, it states the size of its data and location of its bins, and it is located inside the program's heap.

The most straightforward way to determine main arena's location is via *main_arena* debugging symbol. While debugging information is very useful and allows many functions of *HeapAnalysis* to work more reliably, it cannot be assumed that it will always be available. Programs usually have to be compiled in a way that explicitly states the developer's desire for the debugging information to be included to have it end up in compiled code, and practically no program contains debugging information in its binary releases. With open source software it is possible to recompile the program by hand with debugging information enabled, but such a thing cannot be achieved with proprietary programs. Hence, alternative methods of operation are required that do not necessarily rely on debugging symbols.

HeapAnalysis uses two methods to locate the main arena when debugging information is not present. The first method relies on the fact that all arenas are stored in a circular linked list and attempts to locate any additional arenas. If any are discovered and their *ar_ptr* pointers are followed, a pointer into the main heap of the program should eventually be discovered and we can assume its destination to be the main arena. The method locates arenas by looking for memory regions defined by *vm_area_struct* structs. It attempts to interpret their beginning as *heap_info* structs. If a struct is discovered that does not point to a previous *heap_info* struct and specifies its arena as the space right after itself, then the space after it probably really is an arena's *malloc_state* struct, which is temporarily saved and used to locate main arena. If no additional arenas are discovered, that most likely means that the program in questions runs in a single thread and only uses its main arena. In such cases the first method cannot function and the second one is employed instead.

The second method scans the size field of discovered chunks, checking if the *PREV_INUSE* bit is set. If it is not, the previous chunk is most likely freed and should likely (unless it's inside a fastbin) possess a *bk* field with a pointer to previous freed chunk. If these are followed they should lead to main arena. However, as they only lead to bins that belong to the main arena and not its *malloc_state* struct directly, it is still necessary to have a way to precisely locate the

struct. To determine the exact location of main arena's *malloc_state* struct the surrounding memory is scanned, looking for a pointer to the last chunk in the memory area – the *top chunk*. We can check if the chunk that a pointer points to is the last one by summing its size and offset – the result should reach the end of memory area. When the pointer to top chunk is found we have also found the main arena's *malloc_state* struct, since we know where in the struct the pointer resides.

3.2 Locating MMAPED chunks

In comparison to main arena, whose location is set within a limited area and connected to other structures around it, the locations of MMAPPED chunks are harder to determine. They exist simply as chunks in MMAPED areas and belong to no arenas which would link them to other memory structures within the program's heap. Their location is also not limited to a particular area of process space, they could be contained in any unnamed MMAPPED area within it.

The pointers to MMAPPED areas are usually contained on the stack, so scanning the stacks of all threads for pointers and following them could be one way of locating MMAPPED areas, but this would be fairly time consuming and might not contain all MMAPPED areas (such as freed ones). For the lack of a better alternative, the authors of [4] proposed a plausability check, which includes scanning the beginnings of memory regions and checking them against the following criteria in order to determine, whether they could represent MMAPPED chunks:

- The *prev_size* field is not used in MMAPPED chunks and should contain value 0.
- The value in the size field, ignoring the last 3 bits as they are flags, should be at least the size of one page.
- The size value should be divisible by the size of one ppage.
- Summing the chunk's size and its offset should return a value that is still located within its memory region.
- Chunk's address must be divisible by the size of one page.
- The flags (stored in the last three bits of size field) should hold correct values:

PREV_INUSE:	unset
NON_MAIN_AREA:	unset
IS_MAPPED:	set

This method works, but will sometimes not succeed in locating all of the MMAPPED chunks. The reason for this is that while a memory area that is specifically dedicated to MMAPPED chunks does start with a MMAPPED chunk, they may also be placed after other data in an existing memory area. Such chunks are referred to as *hidden* MMAPED chunks and locating them requires the use of different methods. Still, in order to avoid false positives, they are usually not looked for unless there is reason to believe that the information about MMAPPED chunks discovered previously is incorrect.

3.3 Verifying discovered data

Once the data has been located, it is important to make sure that the data makes sense and fits our expectations of what it is supposed to look like. Making sure it does is a part of verifying that our data is indeed chunks and not something else that was picked up alongside them by mistake. Verifying the data is done via a number of sanity checks, that make sure the data is correct. Passing these tests is a good indication that the data was gathered correctly and that there are no issues with it from the heap's perspective. Among them are:

- Checking that the flags of chunks are set correctly, for example that MMAPPED chunks only have the *IS_MAPPED* bit set and that chunks from thread arenas contain the *NON_MAIN_ARENA* bit.
- That the addresses of chunks are aligned correctly, so that they are divisible by the size of a page for MMAPPED chunks and by 8 or 16 (depending on CPU architecture) for regular chunks.
- That the sizes of chunks make sense in terms of alignment and that their size does not exceed the size of their memory area.
- That chunks that are supposed to be allocated aren't parts of any bins

4. CONCLUSION

In this paper we have presented an overview of the GNU C library process heap on Linux operating systems. We explained its structures and how knowing their layout is beneficial for locating data in memory, summarized some of the methods described in [4] that are used to locate important heap components, even when debugging data is not available and then tested the implementations of some of these methods, created as plugins for Rekall memory forensics framework. In doing so we discovered that the newest version of KeePassX is not suitable for analysis with the developed plugin, whereas commands in zsh remain accessible. The analysis with the latest distribution of Rekall was simple and the plugin worked as expected.

5. REFERENCES

[1] Rekall forensics. `http://www.rekall-forensic.com/`. Accessed: 2018-05-10.

[2] The volatility foundation. `http://www.volatilityfoundation.org/`. Accessed: 2018-05-10.

[3] A. Aljaedi, D. Lindskog, P. Zavarsky, R. Ruhl, and F. Almari. Comparative analysis of volatile memory forensics: live response vs. memory imaging. In *Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE SocialCom*, pages 1253–1258. IEEE, 2011.

[4] F. Block and A. Dewald. Linux memory forensics: Dissecting the user space process heap. *Digital Investigation*, 22:S66 – S75, 2017.

[5] F. Block and A. Dewald. Linux memory forensics: Dissecting the user space process heap. Technical Report CS-2017-02, Technische Fakultät, 2017.

[6] S. Davidoff. Cleartext passwords in linux memory. *Massachusetts institute of technology*, pages 1–13, 2008.

Analiza pomnilnika z uporabo generacijskega čistilca pomnilnika

Seminarska naloga pri predmetu Računalniška forenzika *

Jure Jesenšek
Univerza v Ljubljani
Fakulteta za računalništvo in informatiko
Večna pot 113
Ljubljana, Slovenija
jj4001@student.uni-lj.si

Marko Lavrinec
Univerza v Ljubljani
Fakulteta za računalništvo in informatiko
Večna pot 113
Ljubljana, Slovenija
ml9987@student.uni-lj.si

POVZETEK

Analiza glavnih pomnilnikov lahko predstavlja pomemben del forenzične analize. Zaradi razmaha programskih jezikov, ki tečejo v navideznih strojih, pa je potrebno popraviti, oziroma na novo razviti orodja, ki poizkušajo razbrati informacije, vsebovane v njih.
Ta članek se osredotoča na analizo pomnilnika navideznih strojev - natančneje Javanskega navideznega stroja (JVM) HotSpot in pripadajočega čistilca pomnilnika (angl. *garbage collector - GC*). Kljub temu, da je bil nek podatek v programu označen za izbris, pa lahko v pomnilniku ostane še precej časa, saj HotSpot ne briše podatkov za seboj, vendar jih le kopira. S podrobnejšo analizo se je izkazalo, da lahko zaradi tega še vedno pridemo do podatkov, za katere najprej sklepamo, da niso več dostopni.
Ta članek se posveti analizi pomnilnika navideznega stroja HotSpot, vendar bi ga lahko posplošili tudi za druge podobne stroje, kot sta Microsoftov .Net in Googlov V8 JavaScript Engine.

Ključne besede

Forenzika pomnilnika, Java, HotSpot JVM, navidezni stroj

1. UVOD

V tem delu se bomo posvetili pridobivanju podatkov iz glavnega pomnilnika (RAM-a). V ospredju bodo predvsem podatki, ki bi morali biti iz njega že odstranjeni. Tak primer podatkov v Javi predstavljajo podatki, ki so bili domnevno že odstranjeni s čiščenjem pomnilnika (angl. *garbage collection - GC*). Preverili bomo, če se te podatke da obuditi in posledično kakšne informacije nam uspe pridobiti iz njih po izbrisu.

Najprej si bomo ogledali področje in preverili, katere podobne analize so bile že opravljene ter s kakšnimi orodji je moč izvajati forenzično analizo glavnih pomnilnikov.
Nato si bomo ogledali kako poteka shranjevanje in obdelava podatkov v Javanskih programih.
V Poglavju 2 si bomo ogledali pristop k forenzični analizi pomnilnika s programom RecOOp, v Poglavju 3 pa bomo preverili kako se lahko opisano analizo napravi pri preverjanju delovanja zlonamerne kode.

1.1 Pregled področja

Na spletu je moč dobiti več različnih orodij za analizo pomnilnika. Med njimi najbolj izstopata dve orodji.

Prvo pomembnejše orodje je Volatility, za katerim stoji fundacija Volatility Foundation. Orodje je v celoti odprtokodno in objavljeno pod GNUjevo splošno licenco. Podpira pridobivanje podatkov iz glavnega pomnilnika, ki se hrani v sliki diskov ter podpira 32 in 64 bitne operacijske sisteme Windows, Mac OSX in Linux [4].

Drugo pomembnejše in širše poznano orodje je Rekall, za katerim stoji podjetje Google. Tudi to orodje je v celoti odprtokodno in na voljo pod splošnima licencama Apache in GNU. Tudi to orodje je namenjeno preiskavi 32 ter 64 bitnih operacijskih sistemov Windows, OSX in Linux. Orodje izhaja iz orodja Volatility, iz katerega so leta 2013 naredili svoj projekt.[5].

Obe orodji sta namenjeni forenzičnim analizam glavnega pomnilnika, med katerimi lahko iščemo aktivne procese, odprte omrežne povezave in podobno [7].

Obstaja pa tudi nekaj kompleksnejših analiz, kjer so raziskovalci poizkušali iz pomnilnikov mobilnih telefonov pridobiti slike zajete z napravo. V eni od njih so za pridobitev le-teh so uporabili tako imenovano tehniko VCR. Z njo pa so uspeli pridobiti tudi slike, ki se niso nikoli dejansko shranile v shrambo telefona, vendar so bile le prikazane na zaslonu pred zajemom slike [8].

Raziskovalcem je uspelo tudi ugotoviti, da Unix operacijski sistemi in standardne knjižnice včasih neuspešno počistijo pomnilnik za seboj. Podatke procesov, ki še tečejo, pa je možno pridobiti in nadalje analizirati. Iz zajetih podatkov lahko nato napadalci pridobijo izbrisane ključe, objekte ali

*Študijsko leto 2017/2018

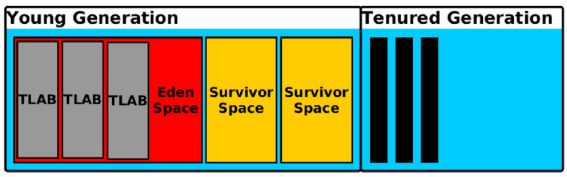
druge podatke [7].

Na posnetku [6] raziskovalka pokaže, kako ji je iz binarnega zapisa podatkov uspelo pridobiti objekte programskega jezika Python, podobno pa bi bilo mogoče narediti tudi za Javo.

1.2 Shranjevanje v pomnilnik

Objekt v Javi se po kreiranju, z namenom optimizacije delovanja, večkrat kopira po pomnilniku.

Ob kreiranju se tako objekt ustvari v rajskem prostoru (angl. *eden space*), ki je razdeljen na lokalne medpomnilnike (TLAB). Splošna hipoteza namreč pravi, da se večina objektov potrebuje le krajši čas - najboljši primer teh so lokalne spremenljivke, ki "živijo" le v obsegu neke funkcije. Kljub temu pa nekateri objekti preživijo čiščenje z GC in so posledično prestavljeni v preživetveni prostor (angl. *survivor space*). S časoma se objekti, ki ostanejo v uporabi, prestavijo v stalni prostor (angl. *tenured generation*). To razvrstitev prikazuje tudi Slika 1, kjer se objekte premika iz leve proti desni.



Slika 1: Postavitev pomnilnika pri Java programu [7].

Zaradi poudarka na hitrosti delovanja se prekopirani podatki na stari lokaciji ne prepišejo na prazno vrednost, ampak tam ostanejo praktično nespremenjeni. Predpostavlja se namreč, da se bodo kasneje prepisali z novimi vrednostmi, vendar pa ni nujno, da se to kdaj dejansko zgodi.

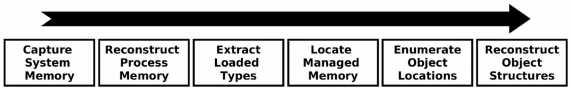
Hitrost izvajanja programa se običajno pohitri, ko računalnik (in posledično JVM) pridobi več glavnega pomnilnika. S tem se namreč pridobi na fleksibilnosti optimizacije prostora na njem, prav tako pa ni več potrebno tako pogosto čiščenj z GC-jem. Ob enem pa se s tem podaljša tudi čas, ko podatki ostanejo v pomnilniku. To pride še posebno prav pri morebitnih forenzičnih analizah, saj se pri tem ohrani in lahko obnovi več podatkov.

JVM prav tako uporablja področja v pomnilniku, ki so namenjena deljenju večjih količin podatkov med knjižnicami napisanimi v programskem jeziku C. To predstavlja še dodatno tveganje, da se bo določen del pomnilnika sprostil, brez da bi bili podatki na njem predhodno uničeni.

2. PRISTOP

Analiza pomnilnika v tem članku se osredotoča na virtualen stroj (angl. *virtual machine*) in dodeljen pomnilnik (angl. *managed memory*). V okviru članka je bilo razvito ogrodje RecOOP, ki uporablja preprosta sistema za interpretacijo struktur v pomnilniku ter dostop do pomnilnika z uporabo navideznih naslovov procesa. Ogrodje je napisano v jeziku Python in podpira so-uporabo z interaktivnimi okolji, kot je IPython [1] in s knjižnicami, kot je Rekall [5].

Na sliki 2 je opisan postopek pridobitve in končne rekonstrukcije objektov iz dodeljenega pomnilnika. RecOOP trenutno deluje nad pomnilnika navideznega stroja HotSpot [2] (podjetja Oracle) na arhitekturi x86. Razširitev delovanja na 64 bitne navidezne stroje in stroje za druge jezike (.Net, JavaScript...) bi zahtevalo le manjše popravke.



Slika 2: Koraki rekonstrukcije objektov.

Orodje RecOOp je bilo testirano na operacijskih sistemih Linux in Windows, in sicer na Ubuntu (32 bit), Windows XP SP3, Windows 7 in Windows 8. Velikost kopice (angl. *heap*) navideznega stroja je bila nastavljena na 2 GiB. Zajete vrednosti v pomnilniku se s pomočjo šablon interpretira v podatkovne strukture. Na različnih operacijskih sistemih nastanejo razlike v količini praznega pomnilnika, ki ga prevajalniki pustijo zavoljo pravilnega razporeda spremenljivk (angl. *field padding*). Na srečo je bilo na ta račun potrebno spremeniti le 8 izmed 150 prej omenjenih C++ šablon.

2.1 Rekonstrukcija procesov

RecOOP začne z rekonstrukcijo procesov. V kolikor procesov pomnilnik še ni bil zajet, ga RecOOP zajame s pomočjo ogrodja Volatility. Proces identificira preko imena ali preko identifikacijske številke procesa (angl. *process ID - PID*), ter oštevilči in pravilno razvrsti okvirje pomnilnika (angl. *memory frame*). Rezultati se nato shranijo v datoteko za nadaljnjo obdelavo ali za pregled z orodji, kot je na primer Radare [3].

2.2 Izločevanje naloženih razredov

Izvajanje poljubnega programa v navideznem stroju se začne z nalaganjem zahtevanih tipov, razredov in ostale kode navideznega pomnilnika. Hkrati se ustvarijo tudi simboli za vsakega od teh elementov. Temu sledi povezovanje (angl. *linking*) kode, razredov in tipov. S tem se zagotovi prisotnost vseh potrebnih tipov in metod, ki se sicer nahajajo v drugih razredih. Po povezovanju in nalaganju (angl. *loading*) pride na vrsto transformacija začetnih razredov (in pripadajočih tipov in programske kode) v strukture, ki so optimizirane za izvajanje v virtualnem stroju.

HotSpot shranjuje potrebne informacije za izvajanje v tri razpršilne tabele (angl. *hash table*) - `SystemDictionary`, `SymbolTable` in `StringTable`. `SystemDictionary` vsebuje informacije o naloženih tipih (razredih). `SymbolTable` vsebuje vse simbole za razrede, metode in spremenljivke. Tabela `StringTable` pa vsebuje znakovne nize (angl. *string*), ki v programu služijo kot konstante, ter nize, ki imajo dolgo življenjsko dobo. V okolje se ponavadi naložijo le tipi, ki so potrebni za povezovanje, kar je uporabno pri razreševanju JAR datotek, katerih vsebina je bila namenoma zakrita (angl. *obfuscated*), saj se s tem zmanjša obseg razredov in tipov, na katere je potrebno biti pozoren.

Orodje RecOOP se najprej osredotoči na simbolno tabelo (angl. *symbol table*), in nadaljuje s sistemskim slovarjem (angl. *system dictionary*). Simbolno tabelo se pregleda na

začetku, ker ima preprosto strukturo, in ker njena vsebina večkrat pride prav v nadaljevanju.

Pri iskanju prej omenjenih tabel si pomagamo z dejstvom, da ti tabeli vedno vsebujeta C++ spremenljivko (lastnost tabele) `_table_size` z vrednostma `0x00004e2b` oziroma `0x000003f1`. Ko sistem najde ti dve vrednosti, poizkusi razčleniti vnose v tabelah. Razpršilne tabele interno vsebujejo polje (angl. *array*) kazalcev (angl. *pointer*) na iskane vnose. Orodje iterira preko polja kazalcev in tolmačiti vnose na naslovih, kamor kažejo kazalci. V kolikor ima vnos pričakovano obliko in vsebuje pričakovane vrednosti, ga interpretira kot veljavnega.

2.3 Identifikacija dodeljenega pomnilnika

Poznavanje lokacije dodeljenega pomnilnika pomaga pri oštevilčevanju (angl. *enumeration*) objektov. Prav tako je koristno pri določanju ali so dobljeni rezultati veljavni. Vendar pa je pravilno klasificiranje dobljenih segmentov zahtevno. Za pravilno identifikacijo objektov se izvede preverjanje tipov nad vsemi ne-primitivnimi tipi v objektu.

Večje število kazalcev na definicije tipov na nekem območju pomnilnika (angl. *type-pointer*, v tem primeru kazalci tipa `Klass*`), lahko nakazujejo lokacijo potencialnih območij v pomnilniku, ki vsebuje dodeljeni pomnilnik. Kazalci tipa `Klass*` so namreč obvezen del vsakega objekta. Veliko število kazalcev na definicije tipov nakazujejo na možnost, da se v temu delu pomnilnika nahajajo objekti. Izjema so področja, ki vsebujejo metapodatke o razredih ter podatkovne strukture, ki jih uporablja prevajalnik. Ker pa so vrednosti na teh območjih poznane, jih je lahko preskočiti. Pri ostalih območjih se zanašamo na preverjanje tipov (angl. *type cheking*) za filtriranje neveljavnih vnosov.

Za lažje določanje mej dodeljenega pomnilnika najprej zanemarimo območja, ki so manjša od 256 KiB, saj je to manj od najmanjše privzete velikosti kopice. Nato iterativno pregledamo strani, ki imajo več kot 10 kazalcev na definicije tipov. Pri vsaki strani število kazalcev popravljamo s premikajočim povprečjem (angl. *moving average*). Za čistilce pomnilnika, ki uporabljajo t.i. *humongous* območja pomnilnika - to so dodelitve velike količine pomnilnika (več MiB) za velike objekte, bi bilo ta algoritem potrebno nekoliko popraviti.

Ta analiza določi meje območja, na katerem bi se lahko nahajala večja koncentracija objektov. Izogibamo se preiskovanju pomnilnika po generacijah (*eden*, *tenured*, itd.), saj bi to lahko prineslo napačne identifikacije. Ker navidezni stroj spreminja velikost kopice glede na trenutno aktivnost programa, bi lahko dva segmenta označili za različna, vendar pa bi bila v resnici nekoč na istem območju.

V primeru, ko je določanje generacij potrebno, si lahko poleg kazalcev na definicije tipov pomagamo še z dnevnikom (angl. *log*) dogodkov čiščenja pomnilnika, ki je del samega navideznega stroja. Vnosi v dnevniku med drugim vsebujejo ime kopice ter začetni in končni naslov. Iskanje z regularnim izrazom (angl. *regular expression - regex*) "`space.*used|Metaspace.*used.`" vrne informacije o praktično vsem dodeljenem pomnilniku.

2.4 Oštevilčevanje in izločevanje objektov

Oštevilčevanje objektov, kot so niti (angl. *thread*), vtičnice (angl. *socket*) in datoteke se izvaja avtomatsko s pomočjo lokacij kazalcev na definicije tipov. RecOOP pa omogoča tudi naknadno oštevilčevanje posebnih tipov objektov tudi po temu, ko določi naslove kazalcev na definicije tipov.

Izločevanje (angl. *extraction*) objektov se začne s preverjanjem, ali naslov ustreza neki osnovni struktiri objekta. Nato s pomočjo informacij o tipih določimo velikost objekta v pomnilniku ter poiščemo njegove reference. Sledi rekurzivno preverjanje vsebovanih ne-primitivnih tipov, pri katerih uporabimo isti postopek. Pri referencah se preverjata tip ter vsebnost vrednosti `null`. Zaradi polimorfizma je potrebno hraniti seznam možnih tipov vsebovanih referenc. Algoritem se zaključi po končanem izločevanju, ter nato nastavi vrednosti spremenljivk (referenc) v objektih. Zaradi preprečevanja neskončne rekurzije se objekti najprej oštevilčijo, nato pa se nastavijo vrednosti spremenljivk. Vrstni red oštevilčevanja in izločevanja objektov je sledeč:

Niti (`java.lang.Thread`) obdelamo najprej. Poleg samega objekta, ki predstavlja nit, pregledamo tudi objekte, ki implementirajo njeno funkcionalnost, v katerih iščemo ostale uporabne informacije. Vsaki niti določimo veljavnost in pregledamo notranje spremenljivke - najpomembnejša med njimi je `eetop`, ki vsebuje naslov niti. S pomočjo tega naslova poiščemo seznam vseh niti v navideznem stroju, ki jih nato prav tako pregledamo z istim postopkom.

Sledi pregled **medpomnilnikov in tokov** (angl. *buffers and streams*), saj so pogosto uporabljeni za prenašanje vhodno/izhodnih podatkov med programom, navideznim strojem in operacijskim sistemom. Delo nam otežuje polimorfizem, saj se pri implementaciji razredov (npr. `java.io.BufferedReader`) uporablja več osnovnih in abstraktnih razredov (npr. `java.io.InputStream`). Da dobimo verigo (oziroma drevo) implementacij je potrebno večkratno preverjanje razmerij med različnimi vrstami razredov in njihovimi implementacijami.

Domače medpomnilnike (angl. *native buffers*) uporablja navidezni stroj za lastno izmenjavo vhodno/izhodnih podatkov. Izkaže se, da ti medpomnilniki implementirajo vmesnik (angl. *interface*) `DirectByteBuffer`, ki podpira direkten dostop do pomnilnika (angl. *direct memory access*). Razredi, ki implementirajo vmesnik `DirectByteBuffer` (npr. `MappedByteBuffer`, `NativeBuffer` in `HeapByteBuffer`) večinoma vsebujejo uporabne vrednosti, ki pa zaradi svoje nestalne narave (angl. *volatility*) pogosto niso zanesljive.

Informacije o datotekah (angl. *file information*) pridobimo s pregledovanjem objektov tipa `java.io.FileDescriptor` ali `java.io.File`. Edina uporabna informacija v teh razredih je ime datoteke in pot do nje (angl. *file path*).

Datoteke JAR vsebujejo informacije o naloženih datotekah in lahko potencialno vsebujejo občutljive informacije o tokovih. Datoteke JAR vsebujejo vse potrebne vire in razrede, potrebne za izvajanje programa ali knjižnice. Razredne datoteke so dekompresirane in naložene zaporedno, zato je pogosto težko pridobiti ostanke informacij, saj le-te hitro izginejo.

Vtičnice razkrivajo informacije o povezavah. Pridobljene informacije lahko obsegajo IP naslove, lokalna in oddaljena vrata (angl. *port*). Prav tako se izvede poizkus povezave vtičnice z znanimi tokovi in medpomnilniki.

Informacije o pod-procesu (angl. *child process*) pridobimo iz razredov `ProcessBuilder` (uporabljen pri tipičnem zagonu procesa) in `Process` (vezan na operacijski sistem). Pri zagonu procesa se objektu poda niz, ki identificira ime procesa. Žal pa se ta niz pri čiščenju pomnilnika pogosto izgubi.
Objekt `Process` ostane precej časa na kopici. Pri testiranju se je tudi po večih iteracijah čiščenja pomnilnika izkazalo, da je prej omenjene objekte še vedno mogoče pridobiti. Prav tako so nekaj uporabnih podatkov vsebovali tudi pripadajoči medpomnilniki `stdout`, `stdin` in `stderr`.
Precejšnja prednost forenzične analize navideznih strojev je deterministično zaporedje dogodkov, saj HotSpot nitim zaporedno dodeljuje pomnilnik. V praksi to pomeni da se objekti, ki so bili zaporedno dodeljeni eni niti, tudi v pomnilniku nahajajo skupaj, ne glede na aktivnosti preostalih niti. To dejstvo pomaga pri določanju razmerij in povezav med dogodki.

3. PRIMERI ANALIZ

V Poglavju 2 je predstavljena zmožnost RecOOP, ki lahko pridobi podatke iz izvajajočih Java programov, na neki sliki diska. Opisan postopek pa se lahko izkoristi tudi za analizo delovanja virusov in škodljive programske opreme. Ta pristop nam pride še posebej prav, če imamo opravka z virusom, ki se samodejno odstrani iz naprave, podatki pa ostanejo v glavnem pomnilniku.

3.1 Virusni program

V okviru članka [7] so raziskovalci na virtualna stroja, ki sta tekla na Linuxu in Widows XP, namestili virus Adwind. Ugotovili so, da se virus med operacijskima sistemoma obnaša nekoliko drugače. Do tega je prišlo, ker je moral virus izrabiti stranska vrata knjižnic operacijskega sistema, te pa se medsebojno nekoliko razlikujejo. Iz obeh sistemov pa je raziskovalcem uspelo priti do objektov, datotek in celo gesel, ki so se nahajali v kopici in jih je virus uporabljal pri svojem izvajanju.

3.2 Virusni posrednik

Raziskovalci so v okviru članka pripravili tudi Javanski program, ki okuži omrežje in na njem postavi posrednik (angl. *proxy*), preko katerega se lahko nato napadalec priključi v računalnik. V takih primerih raziskovalci običajno zelo težko odkrijejo kateri podatki so bili odtujeni in katere ukaze je napadalec pošiljal preko omrežja. Ker pa je bil zlonamerni program spisan v Javi in je izrabljal vtičnice (angl. *socket*) ter predpomnjene podatke, pa so ti ostali v pomnilniku tudi ko niso bili več v uporabi. Primer obnovljenih podatkov vtičnice, pridobljenih iz pomnilnika, prikazuje Slika 3. Na njej se uspe razbrati, da je napadalec pošiljal ukaze "Do something evil".

Ker podatki niso bili nikoli prepisani, je večina napadalčevih podatkov ostala nedotaknjenih v pomnilniku in zato dostopnih za analizo. Izjema so bili tisti, ki so se že prepisali in nekateri podatki, ki so se prebrali preko toka *DataInput-Stream*, te se namreč lahko preberejo tudi neposredno iz vhodne naprave in niso nujno shranjeni v pomnilnik.

Obj. Address	Remote Connection		In/Out	Data (Up to 30 Bytes)
0x91c779b8	10.18.120.18	48002	⇒	Do something evil-48002!
0x91c7ead0	10.18.120.18	48003	⇒	Do something evil-48003!
0x91c85b70	10.18.120.18	48002	⇐	s3cr3t_d4t3_48002-00000000s3cr
0x91c938d8	172.16.124.15	58860	⇒	czNjcjNOX2QOdDNfNDgwMDItMDAw
0x91c980d0	10.18.120.18	48003	⇐	s3cr3t_d4t3_48003-00000000s3cr
0x91ca5cb8	172.16.124.15	58860	⇒	czNjcjNOX2QOdDNfNDgwMDMtMDAw
0x91cbfef0	10.18.120.18	48004	⇒	Do something evil-48004!
0x91cc7008	10.18.120.18	48005	⇒	Do something evil-48005!
0x91ccdee8	10.18.120.18	48004	⇐	s3cr3t_d4t3_48004-00000000s3cr
0x91cdbad0	172.16.124.15	58860	⇒	czNjcjNOX2QOdDNfNDgwMDQtMDAw
0x91ce02c8	10.18.120.18	48005	⇐	s3cr3t_d4t3_48005-00000000s3cr
0x91cedeb0	172.16.124.15	58860	⇒	czNjcjNOX2QOdDNfNDgwMDUtMDAw

Slika 3: Obnovljeni podatki vtičnice [7].

3.3 Skriptni vdor

Raziskovalci so razvili tudi zlonamerno skripto, ki se jo lahko zapakira v vtičnik in naloži na dotCMS[1] strežnik. Izkoristili so administratorske pravice za nalaganje in aktivacijo vtičnika. Ko je bil aktiven, je izkoristil stranska vrata knjižnice wget. Med napadom so raziskovalci večkrat zajeli sliko glavnega pomnilnika in na njej izvedli analizo. Skripti pa je uspelo izvajati tudi sistemske klice, ki so se klicali preko razredov ProcessBulder.

Kljub temu, da so se vmes večkrat izvedla čiščenja z čistilcem pomnilnika, so objekti procesov ostajali v pomnilniku. Poleg objektov pa so v pomnilniku ostale tudi ostale sledi iz katerih je razvidno tudi katere sistemske klice se je izvajalo na sistemu. Ob analizi zadnje slike pomnilnika jim je tako uspelo videti, kaj je počelo 11 od 16 procesov, ki so bili pognani iz skripte.

Ob zaključku analize so raziskovalci preverili še koliko lahko obnovijo s pomočjo orodja Volatility. Med vsemi slikami pomnilnikov jim je uspelo obnoviti le en proces. To kaže na omejenost orodja.

4. NADALJNJE DELO

Pri orodju RecOOP je še kar nekaj možnosti za izboljšave. To je namreč trenutno zelo potratno in med procesiranjem porabi kar veliko računalniških virov. To se še posebej pozna pri več gigabajtnih analizah slik, ali pa ko imamo opravka z več tisoč objekti.

Prav tako je pri nekaterih objektih še vedno nekaj ovir, kar preprečuje da bi se našlo povezave s podatki. Tako je za njih potrebna podrobnejša analiza, ki pa jo program trenutno še ne izvede samodejno.

Orodje trenutno deluje samo na arhitekturi x86 in za analizo Javanskih programov. Dalo pa bi se ga tudi nadgraditi, da bi podpiral 64-bitne operacijske sisteme in druge programske jezike, kot je naprimer .NET.

5. ZAKLJUčEK

Zaradi načina upravljanja pomnilnika, ki je bolj usmerjen k hitrosti kot pa varnosti, lahko z analizo pomnilnika navideznih strojev pridobimo podatke, za katere smo domnevali da niso več dostopni. Za analizo tega pomnilnika je bilo razvito

[1]https://dotcms.com/

orodje RecOOP, ki pa žal ni odprtokodno in prav tako ni na voljo na spletu. Po opisu iz članka, pa lahko sklepamo, da je z njim mogoče doseči več, kot z alternativnimi prosto-dostopnimi rešitvami.

6. VIRI IN LITERATURA

[1] IPython. https://ipython.org//.

[2] Java HotSpot JVM. http://www.oracle.com/technetwork/java/javase/tech/index-jsp-136373.html.

[3] Radare. https://rada.re/r/.

[4] V. Foundation. Volatility. `https://github.com/volatilityfoundation/volatility`, 2018.

[5] Google. Rekall. `https://github.com/google/rekall`, 2018.

[6] Y. Li. Ying Li - where in your RAM is "python san_diego.py"? - PyCon 2015. https://www.youtube.com/watch?v=tMKXcc2-xO8.

[7] A. Pridgen, S. Garfinkel, and D. S. Wallach. Picking up the trash: Exploiting generational gc for memory analysis. *Science Direct*, (20):20–28, Marec 2017.

[8] B. Saltaformaggio, R. Bhatia, Z. Gu, X. Zhang, and D. Xu. Vcr: App-agnostic recovery of photographic evidence from android device memory images. *ACM New York*, pages 12–16, Oktober 2016.

Crashing programs for fun and profit

Fuzzing memory forensics programs

Matej Horvat
University of Ljubljana, Faculty of Computer and Information Science
matej.horvat@guest.arnes.si

ABSTRACT

We review an efficient architecture for testing memory forensics applications with fuzzing, called Gaslight. We present different ways of fuzzing, compare them to Gaslight, present its authors' results, then repeat the experiment. We also present some problems and propose several possible enhancements to the architecture.

1. INTRODUCTION

1.1 About memory forensics

Forensic procedures are typically done on off-line devices: hard disks, turned off mobile phones, etc. However, those procedures can at best only give us a view into what a computing device was used for some time before its seizure and are made harder by encryption and steganography.

Investigators encountering an on-line (running) computing device or computer system have an opportunity to make use of another source of information: the working memory of the device, which may contain data that has not yet been written to persistent storage or that never will be (e.g. encryption keys, passwords, certain caches).

Making an image of an on-line system's memory is harder than making an image of persistent storage for two reasons.

First, as soon as the device is turned off, the memory contents are lost, therefore the memory cannot be (easily) transferred into another device for imaging as is possible e.g. with a hard disk. While it is possible to read the memory of a powered off computer by using a "cold boot" attack as described in [1], it is not always possible or convenient, and is beyond the scope of this paper.

Second, because the image has to be created on the on-line system, not only will the imaging software itself occupy some memory that may have contained valuable data, but in the case of (nowadays standard) multitasking operating systems, other programs will be executing and modifying the memory contents as it is being imaged. The resulting memory image will therefore not contain a snapshot of the memory at one particular point in time, but will rather be a group of snapshots in some interval of time. For example, operating systems and most applications use several data structures that contain pointers to other data structures; if one structure at the bottom of memory points to another structure near the top of the memory, that structure may no longer be there by the time the memory imaging program reads that part of memory because something else replaced it. This is referred to as smearing [2]. Additionally, the device may be infected with malware or be executing a program attempting to destroy evidence before it is collected.

Memory acquisition is not a simple process (and is also beyond the scope of this paper, though interested readers may look at [1], [4], [7]), but what we can do is make sure that memory analysis applications handle images with missing or smeared data correctly. One such approach is fuzzing.

1.2 About fuzzing

Fuzzing is an automated testing technique used to detect how a program responds to invalid or malicious input.

Fuzzing became widely known after it had been used in 1988 by Miller et. al [8] [3] to demonstrate that utilities in several vendors' Unix-based operating systems would crash and possibly execute arbitrary code when given invalid input.

A program that performs fuzzing is called a fuzzer. Examples of fuzzers are AFL and Zzuf [12] [6]. A fuzzer takes one or more valid input files and mutates them in various ways so that all of an application's possible code paths are exercised. It then monitors the execution of the application (somewhat similarly to a profiler) and checks whether the program responded correctly.

A fuzzer may be implemented in several ways:

- As a library that the tested application calls to get fuzzed input and submit statistics.
- With a specialized compiler that automatically inserts calls to fuzzing functions. This is how AFL works [13].
- As a program intercepting I/O operations. This is how Zzuf works [6].

- As a specialized virtual machine intercepting I/O operations and monitoring the execution of the program. AFL also supports this [13].
- It is also possible to fuzz interactive applications, though more customization is required [5].

Each of these approaches has its advantages and disadvantages. A fuzzing library is the most flexible, but requires potentially extensive modifications to the program, which is a problem when one does not have the source code of the application to be tested or the application is written in a programming language that the fuzzer cannot work with. Intercepting I/O operations is very universal, but it is not flexible and may not be able to distinguish between user commands and input data, and virtual machines are powerful but slow.

Programs written in interpreted languages are also problematic for fuzzing because one usually wants to test the interpreted program, not the interpreter, and the interpreter may perform I/O operations other than those required by the program.

Another problem with fuzzing is that it is slow. There are 2^{8N} possible inputs that are N bytes long. With current hardware, it is clearly impossible to test them all and the tested application may only behave incorrectly for a very small fraction of them. AFL's documentation, for example, recommends that test files (used as the starting point to generate fuzzed inputs) be less than a kilobyte in size [13]. Fuzzers that test unmodified applications also typically generate a whole new file (based on the starting input file, but modified) for each execution of the application.

For fuzzing memory forensics applications, this is clearly impossible, as memory images today may be several gigabytes long, so not only would it take forever to fuzz the correct parts of the test file (in the worst case, assuming the fuzzer doesn't know which areas of a file are important), it would also take a long time to make each copy.

However, it is desirable to fuzz memory forensics applications because investigators rely on them to find facts, and a forensics application that crashes or produces incorrect output makes investigation harder. When it is used interactively by an experienced investigator, errors may be recognized easily, but not necessarily if the application is used as part of an automated solution [3].

In this paper, we will look at the fuzzing architecture for memory forensics applications called Gaslight as described and implemented by A. Case, A. K. Das, S.-J. Park, J. Ramanujam, and G. G. Richard in [3].

2. ABOUT GASLIGHT

The authors of Gaslight wanted to create a memory fuzzing architecture with the following properties:

- It should be fast. It should not make copies of memory images, as that would make it too slow for practical use.

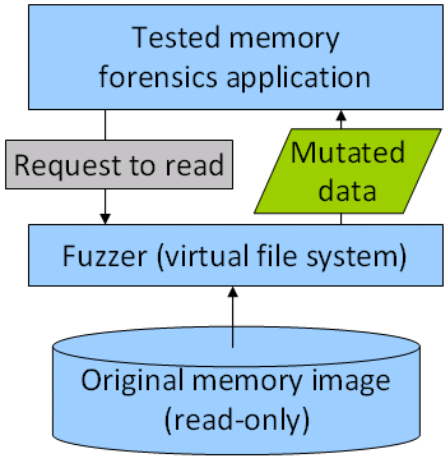


Figure 1: The architecture of the fuzzer implemented as a virtual file system.

- It should be able to test applications without one having to modify them first.
- Because the tested application can need a long time to process one modified memory image, the architecture should be scalable to multiple processors and ideally to multiple computers.
- Errors should be automatically detected.

The authors of Gaslight approached the problem with a virtual file system. It satisfies the first two criteria by letting applications read memory images as usual, but because it is a virtual file system, the driver can modify the data as it is being read, and in a different way each time. This does not require making copies of memory images and it works regardless of whether we are able to modify the tested application or not.

This is somewhat similar to Zzuf, but the advantage is presumably that Gaslight can mutate data in ways that simulate memory smearing and only fuzzes the actual memory image and not user interface elements or other files that may happen to be read by the application (such as libraries and configuration files), which would have to be known in advance and might not be easily determined.

Figure 1 shows the architecture of the fuzzer.

2.1 Virtual file system implementation

The virtual file system was implemented as a FUSE module. Specifically, Fusepy was used, which is a framework for writing FUSE modules in Python [9]. The virtual file system only needs to implement a handful of operations:

- listing the contents of a directory (so that the fuzzed application can find the files it needs),
- getting information about a file,
- opening a file,

- reading data from a file (which is where mutating happens),
- closing a file.

2.2 Usage

Fuzzing is done as follows. First, given a plug-in and a memory image, it must be determined how many read operations the application executes so that it can be known roughly how many times it will have to be run in that configuration. Then, for each mutation type (described below), the application is run with the same configuration the same number of times as the number of reads. The runs differ only by when fuzzing starts; in other words, from which read operation onward the read data is mutated.

Note that the fuzzer does not keep track of the amount of data read, nor where in the file the read data is located. The authors did not explain why they did not consider that important, nor why they mutate data only from one point onward. We can speculate that the reason for the latter is that the number of read operations may change depending on the mutated data, so the counted number of read operations is really only a guideline to determine the number of tests to be run. On the other hand, if the application reads the same data twice (or more times), it can happen because of mutations that the result is not always the same, which would never occur in a real investigation regardless of how badly smeared the memory image was. The authors did not mention this.

2.3 Mutation types

The authors implemented several mutation types that may be applied to a buffer of read data after it has been read from the image but before it has been returned to the calling application:

1. Setting all bytes to 0.
2. Setting all bytes to 255.
3. Setting bytes to random values.
4. Setting every second, fourth, eighth, and 128th byte to a random value.
5. Setting every 2-, 4-, 8-, or 128-byte “page“ to 0, 255, or random values.
6. Incrementing or decrementing every second, fourth, eighth, and 128th byte by 2, 4, 8, 128, or 4096.
7. Incrementing or decrementing every byte in every 2-, 4-, 8-, or 128-byte “page“ by 2, 4, 8, 128, or 4096.

According to the authors, the mutation types were chosen for the following reasons:

- Types 1 to 3 simulate the effects of memory smearing.
- Types 4 and 5 simulate partially overwritten data structures. It is not mentioned whether bytes or pages are counted from the beginning of the buffer or the beginning of the memory image.
- Types 6 and 7 simulate situations where a smeared pointer points to a valid address, but not anymore to the correct data. It is not explained how arithmetic carry or overflow is handled.

The authors of Gaslight primarily focused on testing the stability of various Volatility plug-ins when processing smeared or otherwise corrupted images. They tested a subset of plug-ins that come with Volatility.

3. ABOUT VOLATILITY

Volatility is an open source memory forensics application written in Python [11]. It supports and comes with several plug-ins, which read a memory image and extract various information from it, such as:

- Names of executing processes at the time the image was made, their command line parameters, open files, and working memory contents.
- Most recently executed command line commands and their output.
- Contents of RAM disks and files cached in memory.
- Networking information, such as the ARP cache (known mappings between network and link layer addresses), the local network address, and open sockets.
- Operating system kernel logs.

To extract anything, Volatility must know the exact operating system version used, down to the exact kernel version. The rules used to extract information are specified in operating system profiles. By default, it comes with profiles for several versions of Windows. Mac OS X and Linux profiles are also available [10].

4. RESULTS

Of the subset of Volatility plug-ins that the authors of Gaslight tested, Windows plug-ins never behaved incorrectly. The authors believe this is because they are more mature; they have existed for a longer time and have also been heavily tested in real situations.

Some Mac OS X and Linux plug-ins behaved incorrectly. The most common causes were:

- crashing because of invalid pointers,
- entering infinite loops because of corrupted pointers (e.g. a cycle in a linked list),
- attempting to produce very large outputs because size fields in various data structures were corrupted.

Because it is impossible to truly know whether a program has entered an infinite loop, each test case was given 5 minutes to run. Crashes were detected by scanning the output for words such as ”exception” and ”traceback”, and unusually long outputs were also checked for.

5. REPEATING THE EXPERIMENT

In this section, we will describe how the Gaslight authors' experiments can be repeated and verified, at least partially. We tested a subset of those of Volatility's Linux plug-ins which the Gaslight authors found to be buggy. We assume that they tested Volatility 2.6, which was released in December 2016 and was at the time the article was published the latest official release. At the time of this writing, it still is, though one can always obtain the latest development version from [11], in which the bugs that caused incorrect behavior may have been fixed already. We therefore used version 2.6.

To test it, we implemented a virtual file system with Fusepy on FreeBSD, which, depending on settings, either counted the number of read operations or mutated data with mutation types 1 to 3 as previously described. The other mutation types were not implemented due to their ambiguous descriptions, but we were able to reproduce the results nonetheless.

We used a 128-megabyte memory image of Debian 7.4.0 ("Wheezy") on x86 as the input file to mutate. 128 MB is the minimum amount of memory that Debian requires, and we assumed that a larger image would increase processing time (though we did not test this assumption). The operating system and its specific version was chosen because the authors of Gaslight mentioned using a "32-bit Debian Wheezy" memory image (in addition to others). At the same time, this is the latest version of Debian whose memory images Volatility can analyze, since there are no official profiles available for newer ones.

We did indeed manage to cause Volatility to crash, enter infinite loops, or produce invalid output data that may be easily overlooked if, for example, it was being used as part of a larger analysis suite.

We tested the following plug-ins:

- linux_arp: Shows the ARP table. It executed 23 reads.
- linux_bash: Shows most recently entered bash commands. It executed 1592 reads.
- linux_dmesg: Shows the kernel log. It executed 23 reads.
- linux_library_list: Shows loaded libraries. It executed 654 reads.
- linux_psaux: Shows processes like "ps aux". It executed 238 reads.
- linux_psenv: Shows processes' environment variables. It executed 229 reads.

Figures 2, 3, 4, 5, 6, 7 show the results of fuzzing each plug-in with each mutation type.

Explanation of the terms:

- "Hung" means that Volatility only printed its sign-on message and nothing else. Since test cases were given

Figure 2: Results of fuzzing the linux_arp plug-in.

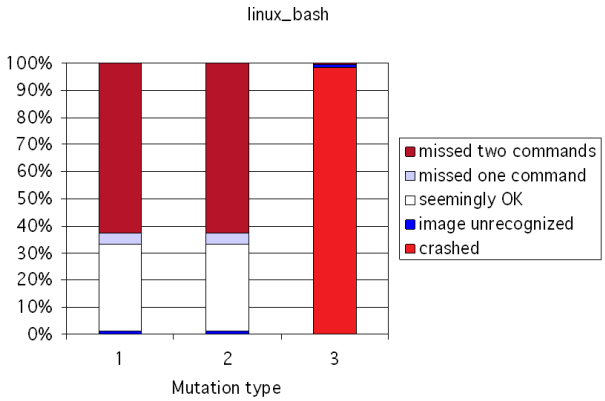


Figure 3: Results of fuzzing the linux_bash plug-in. We executed three commands and then checked how many of them the plug-in could find.

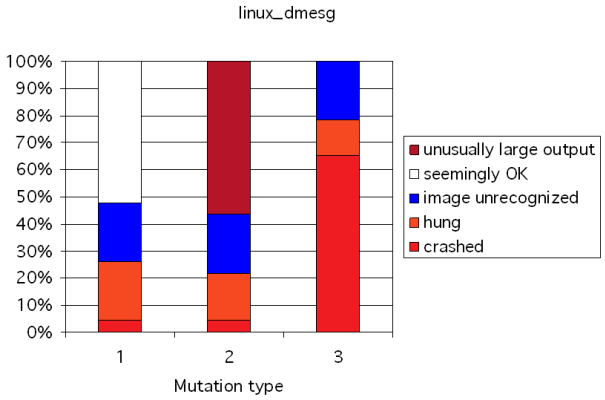


Figure 4: Results of fuzzing the linux_dmesg plug-in.

Figure 5: Results of fuzzing the linux_library_list plug-in.

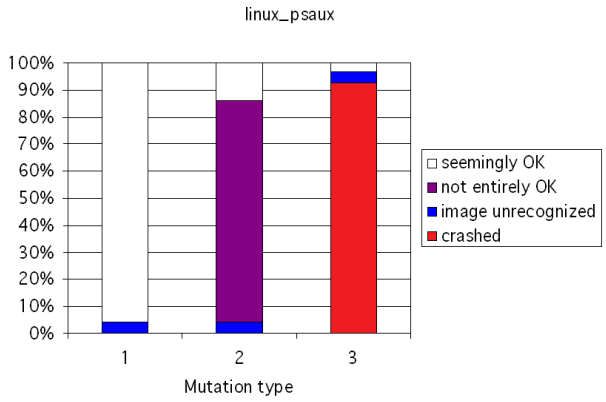


Figure 6: Results of fuzzing the linux_psaux plug-in.

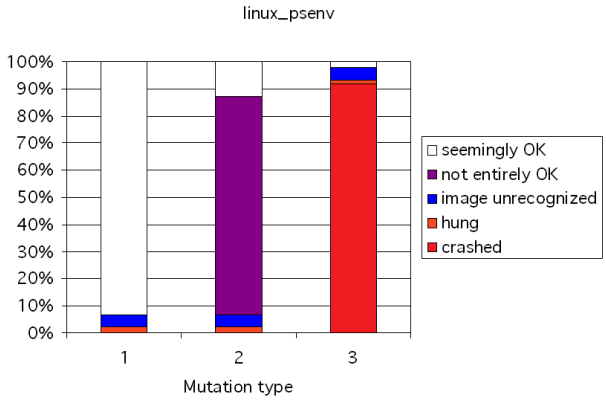


Figure 7: Results of fuzzing the linux_psenv plug-in.

5 minutes to run, it is safe to conclude that the plug-in entered an infinite loop before it could produce any output.

- "Crashed" means that the plug-in caused an uncaught exception which printed a stack trace.
- "Image unrecognized" means that the image was so heavily fuzzed that Volatility did not even recognize it as a valid memory image.
- "Not entirely OK" means that the plug-in recognized that some data was invalid.
- "Seemingly OK" means that we do not think that the plug-in detected any corruption.

Here are some examples of outputs when processing fuzzed data:

linux_arp with mutation type 2 starting at read operation 23 - some IP addresses cannot be obtained:

```
[?
    ] at 33:33:00:00:00:16     on eth0
[?
    ] at 33:33:00:00:00:02     on eth0
[?
    ] at 33:33:00:00:00:fb     on eth0
[?
    ] at 00:00:00:00:00:00     on lo
[?
    ] at 33:33:ff:38:dd:29     on eth0
[224.0.0.22
    ] at 01:00:5e:00:00:16     on eth0
[224.0.0.251
    ] at 01:00:5e:00:00:fb     on eth0
```

linux_arp with mutation type 2 starting at read operation 18 - one MAC address is unusually long:

```
[?
    ] at 33:33:00:00:00:16     on eth0
[?
    ] at 33:33:00:00:00:02     on eth0
[?
    ] at 33:33:00:00:00:fb     on eth0
[?
    ] at 00:00:00:00:00:00:00:
00:00:00:00:00:00:00:00:00:00:00:00:00:
00:00:00:00:00:00:00:00:00:00:00:00 on
    ??????????????????
[?
    ] at 33:33:ff:38:dd:29     on eth0
```

linux_arp with mutation type 2 starting at read operation 15 - some entries are missing:

```
[?
    ] at 33:33:00:00:00:16     on eth0
[?
    ] at 33:33:00:00:00:02     on eth0
[?
    ] at 33:33:00:00:00:fb     on eth0
```

```
[?
    ] at 33:33:ff:38:dd:29     on eth0
```

linux_bash with mutation type 2 starting at read operation 1100 - everything seems normal:

Pid	Name Command	Time
3248	bash	2018-05-09 20:13:18 UTC+0000 echo Hello world!
3248	bash	2018-05-09 20:13:20 UTC+0000 head /proc/cpuinfo
3248	bash	2018-05-09 20:13:22 UTC+0000 uname -a

linux_bash with mutation type 2 starting at read operation 1050 - one command is missing:

Pid	Name Command	Time
3248	bash	2018-05-09 20:13:18 UTC+0000 echo Hello world!
3248	bash	2018-05-09 20:13:20 UTC+0000 head /proc/cpuinfo

linux_bash with mutation type 2 starting at read operation 1000 - two commands are missing:

Pid	Name Command	Time
3248	bash	2018-05-09 20:13:18 UTC+0000 echo Hello world!

linux_psenv with mutation type 3 starting at read operation 125 - crashes because of attempting to read a nonexistant address:

```
Traceback (most recent call last):
  File "/usr/local/bin/vol.py", line 192,
      in <module>
    main()
  File "/usr/local/bin/vol.py", line 183,
      in main
    command.execute()
  File "/usr/local/lib/python2.7/site-
      packages/volatility/plugins/linux/
      common.py", line 64, in execute
    commands.Command.execute(self, *args,
        **kwargs)
  File "/usr/local/lib/python2.7/site-
      packages/volatility/commands.py",
      line 147, in execute
    func(outfd, data)
  File "/usr/local/lib/python2.7/site-
      packages/volatility/plugins/linux/
      psenv.py", line 46, in render_text
    outfd.write("{0:17s} {1:6s} {2:s}\n".
        format(str(task.comm), str(task.pid
        ), task.get_environment()))
  File "/usr/local/lib/python2.7/site-
      packages/volatility/plugins/overlays/
      linux/linux.py", line 2149, in
      get_environment
    argv = proc_as.read(start, self.mm.
        env_end - self.mm.env_start + 10)
  File "/usr/local/lib/python2.7/site-
      packages/volatility/addrspace.py",
      line 276, in read
    return self._read(addr, length, False)
  File "/usr/local/lib/python2.7/site-
      packages/volatility/addrspace.py",
      line 258, in _read
    data = read(paddr, datalen)
  File "/usr/local/lib/python2.7/site-
      packages/volatility/plugins/
      addrspaces/standard.py", line 104, in
       read
    data = self.fhandle.read(length)
IOError: [Errno 14] Bad address
```

The source code of the virtual file system and the scripts used to perform fuzzing and analysis are listed in the appendices, as well as the specific steps to be taken to install the required software.

6. CONCLUSION

We have shown that fuzzing is a valid way of testing memory forensics programs and that the specific methods proposed and implemented by Gaslight's authors really do work (even though their descriptions were not always entirely unambiguous). They are not in fact limited to testing memory forensics programs, but could be used to fuzz other kinds of programs that usually work with large input files.

To be practical, the fuzzer would also need to support easy distributed operation, which is something that neither we nor they implemented yet. As an example, we needed a few dozen hours to run all our tests, though this mostly depends on the number of processors and their speed (Volatility, for example, is single-threaded and CPU-bound but does not require a significant amount of memory, so multiple instances can be easily run at once).

However, detecting that a program crashed is only half the solution. The developer still has to manually examine the stack traces to find what exactly caused the program to crash (in the case of infinite loops, this is even harder). It would be more convenient if the fuzzer could parse stack traces and group together test cases that expose the same problem.

7. REFERENCES

[1] R. Carbone, C. Bean, and M. Salois. *An In-depth Analysis of the Cold Boot Attack: Can it be Used for Sound Forensic Memory Acquisition?* Valcartier:

Defence Research and Development Canada, 2011.
[2] H. Carvey. http://seclists.org/incidents/2005/Jun/22, 2005.
[3] A. Case, A. K. Das, S.-J. Park, J. R. Ramanujam, and G. G. Richard. Gaslight: A comprehensive fuzzing architecture for memory forensics frameworks. *Digital Investigation*, 22:S86 – S93, 2017.
[4] R. M. Forensics. Rekall forensics blog: The pmem memory acquisition suite. http://blog.rekall-forensic.com/2015/04/the-pmem-memory-acquisition-suite.html, 2015.
[5] A. Hertzfeld. Monkey lives. http://www.folklore.org/StoryView.py?story=Monkey_Lives.txt.
[6] S. Hocevar. Zzuf. http://caca.zoy.org/wiki/zzuf.
[7] R. W. McGrew and R. Schramp. msramdmp. https://github.com/Schramp/msramdump, 2015.
[8] B. P. Miller, L. Fredriksen, and B. So. An empirical study of the reliability of unix utilities. https://fuzzinginfo.files.wordpress.com/2012/05/fuzz.pdf, 1988.
[9] Various. Fusepy. https://github.com/fusepy/fusepy.
[10] VolatilityFoundation. Volatility profiles for linux and mac os x. https://github.com/volatilityfoundation/profiles.
[11] VolatilityFoundation. Volatitlity: An advanced memory forensics framework. https://github.com/volatilityfoundation/volatility.
[12] M. Zalewski. American fuzzy lop. http://lcamtuf.coredump.cx/afl/.
[13] M. Zalewski. American fuzzy lop: readme file. http://lcamtuf.coredump.cx/afl/README.txt, 2016.

APPENDIX

A. SETTING UP THE EXPERIMENT

To obtain the memory image:

1. Start Debian 7.4.0 in a virtual machine such as VirtualBox. The fastest way is to use a "live CD": http://ftp.riken.jp/Linux/debian/debian-cdimage/archive/7.4.0-live/i386/iso-hybrid/debian-live-7.4-i386-standard.iso
2. When booting, choose the i686 kernel with PAE, otherwise Volatility will not recognize the memory image as belonging to this operating system.
3. Log in and execute a few commands, such as:

```
echo Hello world!
head /proc/cpuinfo
uname -a
```

4. Create the memory image. For VirtualBox, refer to: https://www.virtualbox.org/ticket/10222

Of course, one could also obtain an image from a physical machine, with real smearing, with methods described in [1], [4], [7]. There is probably no advantage in doing so as we mutate the data anyway.

To install Volatility and Fusepy on FreeBSD 11.1:

1. If the ports tree is not installed yet, install it:

```
portsnap fetch extract
```

2. Compile and install Volatility (version 2.6 at the time of this writing):

```
cd /usr/ports/security/py-volatility
make
make install
```

Note: it can take several hours to compile and install all required dependencies.

3. Get the additional operating system profiles and copy the one for Debian 7.4.0 (x86) to where Volatility can find it:

```
git clone https://github.com/
    volatilityfoundation/profiles
cp profiles/Linux/Debian/x86/Debian74.
    zip /usr/local/lib/python2.7/site-
    packages/volatility/plugins/overlays
    /linux/
```

Volatility is now ready to use.

4. Compile and install Fusepy:

```
git clone https://github.com/fusepy/
    fusepy
cd fusepy
python setup.py install
```

5. Install the FUSE libraries and load the required kernel module:

```
pkg install fusefs-libs
kldload fuse
```

We can now use the fuzzer.

B. VIRTUAL FILE SYSTEM SOURCE CODE

This is the file Fuzzer.py, which implements the virtual file system. When run, it creates the virtual file system (which "passes through" to a real directory, but mutates data when it is read according to which mutation type is chosen) in and mounts it in the directory specified on the command line. It then waits for I/O requests and responds to them. It is based on the "loopback.py" example that comes with Fusepy.

```
from __future__ import print_function,
    absolute_import, division

import logging

from errno import EACCES
from os.path import realpath
from sys import argv, exit
from threading import Lock

import os
import random

from fuse import FUSE, FuseOSError,
    Operations, LoggingMixIn
```

```
class Fuzzer(LoggingMixIn, Operations):
  def __init__(self, argv):
    # Initialize the fuzzer. This is called
        when mounting the file system.

    self.RootPath = argv[2]
    if self.RootPath[-1] != "/":
      self.RootPath += "/"

    self.Label = argv[3]

    self.Mutation = int(argv[4])
    if self.Mutation != 0:
      self.FuzzFrom = int(argv[5])

    self.RWLock = Lock()
    self.ReadCount = 0

  # The I/O operations that we do not
      implement:
  chmod = None
  chown = None
  create = None
  flush = None
  fsync = None
  link = None
  getxattr = None
  mkdir = None
  mknod = None
  readlink = None
  rename = None
  rmdir = None
  statfs = None
  symlink = None
  truncate = None
  unlink = None
  utimens = None
  write = None

  def access(self, path, mode):
    if not os.access(self.RootPath + path,
        mode):
      raise FuseOSError(EACCES)

  def getattr(self, path, fh=None):
    st = os.lstat(self.RootPath + path)
    return dict((key, getattr(st, key)) for
        key in (
      "st_atime", "st_ctime", "st_gid", "
          st_mode", "st_mtime", "st_nlink",
      "st_size", "st_uid"
    ))

  def open(self, path, flags):
    if flags & (os.O_WRONLY | os.O_CREAT |
        os.O_TRUNC):
      # Forbid opening for writing.
      raise FuseOSError(EACCES)

    return os.open(self.RootPath + path,
        flags)

  def read(self, path, size, offset, fh):
    with self.RWLock:
      os.lseek(fh, offset, 0)
      Result = os.read(fh, size)

      if self.Mutation == 0:
        self.ReadCount += 1
      elif self.ReadCount >= self.FuzzFrom:
        # Mutate the data.
        if self.Mutation == 1:
          Result = chr(0) * len(Result)
        elif self.Mutation == 2:
          Result = chr(0xFF) * len(Result)
        elif self.Mutation == 3:
          L = len(Result)
          Result = ""
          for i in range(L):
            Result += random.randrange(256)

      return Result

  def readdir(self, path, fh):
    return os.listdir(self.RootPath + path)

  def release(self, path, fh):
    if self.Mutation == 0:
      # Save the number of reads.
      ReadCountFile = open(self.RootPath +
          self.Label + "-NumReads.txt", "w"
          )
      ReadCountFile.write(str(self.
          ReadCount))
      ReadCountFile.close()

    return os.close(fh)

if __name__ == "__main__":
  if len(argv) < 5:
    print(
      "Parameters:\n" + \
      "To count: mountPoint rootDir label
          0\n" + \
      "To fuzz:  mountPoint rootDir label
          mutationType fuzzFrom"
    )
    exit(1)

  logging.basicConfig(level=logging.DEBUG)

  # Set foreground to True to log all I/O
      operations.
  fuse = FUSE(Fuzzer(argv), argv[1],
      foreground=False)
```

C. TEST SCRIPTS

First, we count how many read operations each plug-in executes:

```
#!/bin/sh

mountPoint=$1

for i in arp bash dmesg library_list psaux
    psenv; do
```

```
  python Fuzzer.py $mountPoint $(pwd)
      linux_$i 0
  vol.py linux_$i -f $mountPoint/
      DebianWheezy.ram --profile=
      LinuxDebian74x86 > /dev/null
  umount $mountPoint
done
```

Then, we can start fuzzing. The following script takes the names of plug-ins as parameters and fuzzes them with mutation types 1 to 3, starting with a different read operation each time:

```
#!/bin/sh

mountPoint=$1
resultDir=$2
shift 2

mkdir -p $resultDir
for i in $*; do
  for j in $(seq 1 3); do
    for k in $(cat linux_$i-NumReads.txt);
        do
      python Fuzzer.py $mountPoint $(pwd)
          linux_$i $j $k
      OutName=$resultDir/$i-$j-$k
      timeout 300 vol.py linux_$i -f
          $mountPoint/DebianWheezy.ram --
          profile=LinuxDebian74x86 >
          $OutName 2> $OutName
      umount $mountPoint
    done
  done
done
```

To make use of multiple processors, we just execute multiple instances of the script at the same time, each with a different mount point and list of plug-ins (but not necessarily a different result directory).

Some plug-ins execute much more read operations than others, so it is recommended to look at the read counts first to decide how we are going to distribute the processing.

To make use of multiple computers (which we did not), we would need to install the same software on all of them and then again execute the script with different parameters on each, possibly using a network share.

This is, of course, all very primitive. An obvious enhancement would be to use a queue to optimally assign work to members of the cluster.

D. ANALYSIS PROGRAM

This program reads the results and outputs them in the CSV format for use in tables or charts.

```
import os

Results = {}
for i in os.listdir("."):
  if "-" in i:
    Plugin, Mutation, FuzzFrom = i.split("-
        ")

    if Plugin not in Results:
      Results[Plugin] = {}
    if Mutation not in Results[Plugin]:
      Results[Plugin][Mutation] = {}

    Size = os.stat(i).st_size
    File = open(i, "r")
    Data = File.read()
    File.close()

    if Size == 47:
      # The file only contains the
          Volatility sign-on message.
      Verdict = "hung"
    elif "Traceback" in Data:
      # Python stack trace.
      Verdict = "crashed"
    elif Size > 0x10000:
      Verdict = "unusually␣large␣output"
    elif "No␣suitable␣address␣space␣mapping
        ␣found" in Data:
      # Volatility couldn't recognize the
          image because it was too
          corrupted.
      Verdict = "image␣unrecognized"
    else:
      if Plugin == "bash" and "cpuinfo" not
          in Data:
        # Found "echo", but not the other
            two commands.
        Verdict = "missed␣two␣commands"
      elif Plugin == "bash" and "uname" not
          in Data:
        # Found the first two commands, but
            not "uname".
        Verdict = "missed␣one␣command"
      elif "????" in Data:
        # Volatility recognized some data
            as invalid.
        Verdict = "not␣entirely␣OK"
      else:
        Verdict = "seemingly␣OK"

    if Verdict not in Results[Plugin][
        Mutation]:
      Results[Plugin][Mutation][Verdict] =
          0
    Results[Plugin][Mutation][Verdict] += 1

# Print a CSV table for each plug-in.
for Plugin in Results:
  print("linux_" + Plugin)
  Mutations = sorted(Results[Plugin])
  print(";" + ";".join(Mutations))

  # Gather all verdicts for this plug-in.
  Verdicts = set()
  for Mutation in Results[Plugin]:
    Verdicts = Verdicts.union(Results[
        Plugin][Mutation])
```

```
for Verdict in sorted(Verdicts):
  Buffer = Verdict

  for Mutation in Mutations:
    All = sum(
      [Results[Plugin][Mutation][i] for i
           in Results[Plugin][Mutation]]
    )

    try:
      Num = Results[Plugin][Mutation][
          Verdict]
    except KeyError:
      Num = 0

    Buffer += ";" + str(Num * 100.0 / All
        )

  print(Buffer)
```

Keystroke dynamics features for gender recognition

Ajda Lampe
Univerza v Ljubljani
Fakulteta za računalništvo in informatiko
Ljubljana, Slovenia
al3318@student.uni-lj.si

David Klemenc
Univerza v Ljubljani
Fakulteta za računalništvo in informatiko
Ljubljana, Slovenia
dk0189@student.uni-lj.si

ABSTRACT

In digital forensics, one often finds himself in a position where the ability to profile the computer user may simplify the task of finding a suspect. Most of the research in the field focuses on recognizing the gender or age which are two of the most informative characteristics and usually the first ones a digital forensic wants to know. The ways to do so vary from using complex ones such as voice recordings, images, signatures to fairly simple ones like the way a person types. This field of study is called keystroke dynamics. Authors of the reference paper chose to predict a person's gender based on keystroke dynamics, since this is, as opposed to the pictures, a non-intrusive method. They assembled a dataset, recording users during daily computer usage, calculated features and lowered the dimensionality of data and finally trained a few of the most popular classifiers for this binary classification task. Using a radial-basis function network (RBFN), they achieve the highest accuracy reported in the field to date.

Keywords

keystroke dynamics, gender recognition, digital security, behavioral biometrics, machine learning

1. INTRODUCTION

Keystrokes are just one of the so-called forensic behavioral biometrics, others being the way of walking, voice, signature and similar. Biometric technologies based on user's behavior have an advantage over the traditional ones in enabling one to collect data without interfering with user's work by constantly asking for their consent or cooperation and usually does not require any additional hardware. Collected data is most frequently used to predict a person's age and gender, since those are two of the most important characteristics and also most commonly lied about by malicious users. This work is focusing on the way a person types, which is referred to as keystroke dynamics.

The basis of this seminar work is a paper written by Tsimperidis et. al. [10]. They assembled a dataset by recording volunteers during their daily usage of the computer. This so-called free-text mode enabled them to get data as close to their natural typing dynamics as possible. They then generated features from the data, using the down-down and up-down latencies, which are time between pressing two consecutive keys and time from when the key was pressed until it was released, respectively. This resulted in over 10,000 features when using a standard computer keyboard. To avoid long training times, they selected only a few hundred features and used them for training five distinct classifier models - naive Bayes (NB), support vector machines (SVM), random forests (RF), multi-layer perceptron (MLP) and radial basis function network (RBFN) with the goal of proving robustness of the features. The latter one achieved state-of-the-art results in the field of gender recognition with keystroke dynamics.

The remainder of this paper is organized as follows. Section 2 overviews the related work, Section 3 describes the assembly of the dataset and the methods used to predict gender, Section 4 presents the experimental evaluation, and finally, conclusions are drawn in Section 5.

2. RELATED WORK

Use of biometrics is a very common approach in various fields related to computer security and digital forensics. The research from other fields such as online marketing can also be well transferred to the field of interest since the objective is usually similar - to characterize an unknown user from available information. Biometrics include speech, image, video, gait, keystrokes, signature and others.

Using facial images for estimating user's gender and age is popular, due to large amounts of such data being used in everyday life. Eidinger et. al. [5] found there was lack of facial image data labelled for age and gender recognition since most of the research is going in the direction of face recognition. They assembled a dataset with images taken under challenging conditions, taken with smart-phones and other devices and use dropout principle combined with SVM to prevent overfitting. Kalansuriya et. al. [7] used neural networks for a similar task, focusing on images taken under various illumination conditions. [2] on the other hand, focuses on using color-based features extracted from social media profiles to predict gender without being influenced by a language. Barkana and Zhou [3] use pitch range features

in their work to predict age and gender of a speaker using kNN and SVM classifiers. Their idea was to construct features that capture pitch changes over time. They achieve the most notable accuracy of 92.86% for middle-aged females, while the worst accuracy corresponds to young males. Human gait can also be effectively used as a way to recognize gender, as is shown by Li et. al. [8]. They detect seven parts of the human body - head, arm, trunk, thigh, front leg, back leg and feet - to capture person's motion and use it to predict gender.

Another behavioral biometric is the way a person types and respective area of research is referred to as keystroke dynamics. Its advantage over above mentioned is that no additional hardware, like camera or microphone, is needed to get accurate data. Many attempts were made to use keystroke dynamics as a supplement to regular login credentials like username and password, showing that there is a lot of potential in the field. As argued by Bartlow and Cukic [4], credentials are weak protection, while fingerprint and face recognition are not always feasible on remote systems and require specific hardware. They propose to join the regular username + password authentication with keystroke dynamics. They report a decrease in system penetration rate by 95% to 99%. Al-Jarrah and Mudhafar [1] use a similar idea to distinguish between a true user and impostor from the way of typing a password. They used key-press duration and digram latency features to create a vector of medians and standard deviations for each feature. To authenticate a user, they compare the test sample with a median vector and count number of features that differ by less than standard deviation and accept the user as genuine if the sum is larger than some threshold. Monaco et. al. [9] attempt to continually authenticate a user. They claim the ability to verify a user from roughly 1 minute of keystroke input, which makes up for a fast intruder detection. They achieve 99% accuracy when training on 14 participant samples and 96% on 30 participant samples, which shows that this direction is promising for further research.

In addition to the keystroke data, Giot and Rosenberger ([6]) use the keystroke patterns to predict gender and thus increasing accuracy of user authentication. Training on data from multiple users typing the same password, they report 91% gender prediction accuracy on the test dataset. Combining the new information with the keystroke patterns, they achieve 20% gain on the task of user authentication. Tsimperidis et. al. [11] focus only on gender identification using keystroke duration features. In an attempt to show that such features are language independent, they use data from people typing in different languages and achieve 70% accuracy. As an upgrade, Tsimperidis et. al. recently published another paper ([10]) in this category, which serves as a basis for this seminar and is described in detail in the following sections.

3. METHODOLOGY

Authors of [10] split their work into three parts, corresponding to the subsections of this section. First they had to acquire data, then transform it into a form that is compatible with the classification algorithms, and lastly, train the classifiers and evaluate the results.

Table 1: Numbers of log files returned per age, educational level, and daily usage in hours.

Characteristic	Class	Female	Male
Age	18-25	11	21
	26-35	66	36
	36-45	39	54
	46+	7	14
Educational level	USCED-3	11	29
	ISCED-4	6	9
	ISCED-5	24	19
	ISCED-6	52	33
	ISCED-7-8	30	35
Daily usage in hours	0-1	15	8
	1-2	19	27
	2-4	23	31
	4-6	13	25
	6+	53	34
Total		123	125

3.1 Collecting data

To create the dataset, they first needed volunteers who would agree to install a key-logger on their computer and send the logs back to the researchers. For that purpose, they addressed a few hundreds of people, among which only 75 returned the log files. On average, they received 3 to 4 log files per participant. The distribution of log files according to different user characteristics is shown in Table 1, which implies that the participants are well distributed and balanced between genders.

The method of collecting the data during participant's daily usage of computer is referred to as "free-text" mode, the other mode being "fixed-text" where the participants are given a text they are supposed to type. The "free-text" mode is usually preferred, since it is less intrusive for the user and enables the dynamics to be as close to the natural one as possible, but can present some privacy issues, since it can reveal user's password or other private information. The "fixed-text", on the other hand, lets researchers focus on particular features of interest and secure private data, but as a result, the dynamics are not necessarily representative of each user.

The key-logger that participants had to install, generates a comma-separated-value output file, where each press of a key consists of two events labeled "dn" and "up" for key press and key release, respectively. Each line represents a user's action. The first field is a virtual key code of the pressed key, the second one is a date in yyyy-mm-dd format, the third represents a number of milliseconds since the start of that day, namely 00:00, and in the last one is the label "up" or "dn" of the action. An example of the output is shown below:

```
42, #2018-05-08#, 29157674, dn
30, #2018-05-08#, 29157809, dn
42, #2018-05-08#, 29157895, up
30, #2018-05-08#, 29157935, up
33, #2018-05-08#, 29158079, dn
33, #2018-05-08#, 29158166, up
20, #2018-05-08#, 29158357, dn
20, #2018-05-08#, 29158451, up
```

The number of features that can be derived is enormous, but the authors of the paper focus on two different types of features. First one is called down-up latency, which is the time elapsed from the moment a key was pressed until the moment it was released. This yields n features, which equals a number of keys on the keyboard. The second type of features is down-down latency, which is the time elapsed between presses of two consecutive keys. Calculating the latency for each pair of keys on the keyboard, called digram, yields another n^2 features. The total number of features extracted then equals $n+n^2$ which would cause the training process to be slow.

3.2 Feature selection

To extract the features from received log files, they developed another software, that calculates average values of keystroke durations and down-down latencies. For each log, it only counts the keys that appear at least 5 times and the digrams that appear at least 3 times, in order to avoid the outliers, and treat the rest as missing values. This results in over 10,000 features on an average computer's keyboard, so it's very important to have means of selecting only the most informative ones.

The entropy $\mathrm{H}(x)$ plays a great role in deriving the system for selecting the best features. Its equation is as follows:

$$\mathrm{H}(x) = -\sum_{i=1}^{m} P(x_i) \log P(x_i), \tag{1}$$

where m is the length of vector x and $P(x_i)$ is a probability of x_i. In the case of gender classification, m equals 2 and the probability equals probabilities of male or female in the whole population, which are around 0.5. The entropy is then around 0.7. To find the most informative features, we calculate information gain (IG) for each of them, which is given by:

$$\mathrm{IG}(x, feature) = \mathrm{H}(x) - \mathrm{H}(x|feature), \tag{2}$$

where

$$\mathrm{H}(x|feature) = \frac{1}{N} \sum_{j=1}^{k} n_j \, \mathrm{H}(x_j), \tag{3}$$

N being the number of instances in dataset and k is the number of groups, according to the value of a feature, into which the dataset is split. The features with the highest IG value are then selected. The analysis shows that the three most informative features are N-A (time between press of key N, followed by press of key A), M-O and K-A latencies. The most informative keystroke durations are those of keys A, D and W, which ranked 7, 9 and 17 on the list, respectively.

3.3 Model evaluation

Authors trained 5 different models on the dataset, assembled as described in previous sections. To verify and compare them correctly, they applied some standard machine learning procedures. First, they use cross-validation to validate the models and prevent overfitting to training data and then evaluate them using some well-known model quality measures.

3.3.1 Cross-validation

Cross-validation is one of the standard machine learning procedures for training models. The dataset is split into k folds of approximately the same size and then in each iteration, a model is trained using all folds but one, each time leaving out a different fold to be used for evaluation of the model. This approach is especially useful when the number of samples is relatively small, so one cannot afford a large enough validation set that would make the measurements reliable. At the same time, comparing results of models trained on different folds can serve as an evaluation of the model robustness. Additionally, because the dataset is small, the computational complexity of the training process using cross-validation is not problematic.

Authors of the paper used 10 folds, each containing 24 or 25 files. This enables them to put all of the files from one user into the same fold, thus preventing overfitting by testing on data very similar to the training one, which would consequently reduce the reliability of the measures.

3.3.2 Quality measures

To evaluate the models, they used multiple quality measures. Among the most common and intuitive is classification accuracy (CA), the ratio of correctly classified samples, which gives a nice idea about the accuracy of the model but does not take into account the possibly uneven distribution of false positives and false negatives. To gain a better insight, they also calculated the F-score (F), which is defined as a harmonic mean of precision and recall:

$$\mathrm{F} = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}, \tag{4}$$

where precision is defined as a fraction of true positives among all predicted positives and recall as a fraction of true positives among all samples labeled as positive. To give idea on time complexity of each model, the CPU time needed to train a classifier (TBM) is measured.

4. EXPERIMENTAL EVALUATION

The first feature that one thinks of when talking about keystroke dynamics is most likely typing speed. The fact that there appears to be no research in the field, focusing on it is an indication that it is most likely not a very informative feature. Despite that, the authors calculated statistics for each gender to find out if it's possible to distinguish a gender solely on the typing speed. The results show that the average time between consecutive key presses is 373.04 ms with a standard deviation of 135.26 for males and 375.71 ms with 116.86 ms standard deviation for women. The average values are fairly close, while the standard deviations are relatively large, which implies that typing speed by itself is not a very informative feature for predicting a person's gender.

For that reason, they used five machine-learning models, capable of extracting patterns that may be less intuitive to a human. The classifiers used are support vector machines (SVM), random forest (RF), naive Bayes (NB), multilayer perceptron (MLP) and radial basis function network (RBFN). The models were trained on different subsets of features and with different model parameters and evaluated using measures, described in Subsection 3.3, to find the optimal number of most informative features. They used between 50 and 400 features, with a step of 50.

Table 2: Classification accuracies and times to build a model for all 5 models for a different number of features. Highest CA for each classifier is marked in bold.

# of feats.	SVM		RF		NB		MLP		RBFN	
	CA	TBM	CA	TBM	CA	TBM	CA	TBM	CA	TBM
50	73.8%	0.16	77.0%	1.00	69.0%	0.03	73.8%	8.55	81.9%	0.53
100	81.9%	0.13	79.8%	2.65	77.0%	0.02	81.9%	31.33	88.3%	0.73
150	**85.1**%	0.16	80.7%	2.03	77.4%	0.18	**85.1**%	73.47	92.7%	2.43
200	83.9%	0.19	78.2%	4.60	77.0%	0.02	84.7%	120.43	93.2%	2.95
250	84.3%	0.22	**81.9**%	6.65	**78.6**%	0.09	82.7%	181.93	93.2%	3.68
300	84.7%	0.33	80.7%	6.15	76.6%	0.08	81.9%	274.20	94.3%	4.31
350	**85.1**%	0.31	80.7%	8.22	76.6%	0.02	83.9%	373.90	**95.6**%	4.89
400	84.7%	0.42	79.0%	8.14	76.6%	0.02	79.8%	509.65	94.8%	5.46

To train the SVM model, they used the polynomial kernel (Polykernel), which gave the best experimental results, compared to RBF Kernel, string kernel, PUK kernel and normalized Polykernel. They also found out that the performance of the classifier is not very dependent on the value of parameter C - a regularization parameter that defines a degree to which a margin can be violated. Using 200 features, for example, the training accuracy is persistently over 80% for C values from 1 to 15. Similar results are obtained when training with other numbers of features. The accuracy of the SVM model reaches maximum values when trained with 150 and 350 features, but does not change drastically when using any number of features higher than 100. The RF classifier is similarly persistent for various numbers of features and is not greatly influenced by the number of trees. For example, with 100 features used, the numbers of trees between 140 and 400 result in only about 2% difference in accuracy. Similar behavior is observed with RBFN and MLP classifiers. The classification accuracy and time, necessary to build a model, are shown in a Table 2.

Other accuracy measures turned out to be highly correlated with the CA value, so they are not specifically shown in the table, but can be seen in the original paper [10]. The F-measure is mostly the same as classification accuracy or deviates by at most 0.002.

From Figure 1 we can observe that the RBFN significantly outperforms the rest of the classifiers regardless of the number of features. On the other hand, the naive Bayes classifier always achieves the lowest accuracy. Additionally, Figure 2 shows that the naive Bayes takes very short time to train, while the time grows significantly for the multilayer perceptron with increasing number of input features. This is due to a great increase in the number of weights that need to be trained and consequently the number of iterations needed to train the model.

Results imply that the best performing model is RBFN and it's time complexity is not greatly influenced by the dimensionality of the training data. Following are multilayer perceptron and the support vector machine. MLP gives CA above 80% when trained on at least 100 features, but its training time grows quickly with the number of features. SVM, on the other hand, appears to result in the best trade-off between CA and TBM, since its computational complexity is independent of the number of features. The random forest is slightly less accurate but has a big advantage of being more easily interpretable than others and can help us understand which feature values really distinguish between genders. The naive Bayes classifier looks much worse than the rest but is really fast in terms of training time, which is is not affected by the training set's dimensionality and does not require any parameters to train a model.

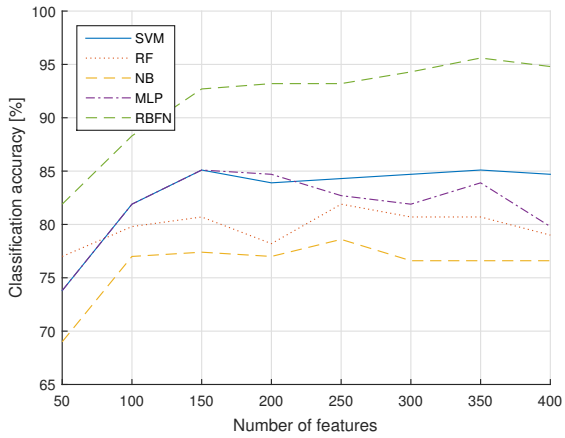


Figure 1: A plot of classification accuracies with respect to number of features for each of the five classifiers.

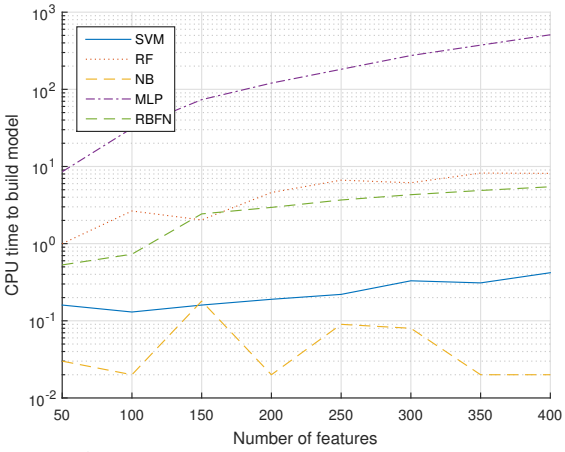


Figure 2: A plot of TBM with respect to number of features for each of the five classifiers. The y-axis is in logarithmic scale.

However, the objective of the paper was not so much to find the best performing classifier for the task, but rather to study a possibility to robustly predict a gender based on the extracted features and the number of features needed to give accurate predictions. It turned out that the CA is not heavily influenced by the number of features used. For the range of 150 to 350 features the CAs are $84.5 \pm 0.6\%$ for SVM, $80.0 \pm 1.8\%$ for RF, $77.6 \pm 1.0\%$ for NB, $83.5 \pm 1.6\%$ for MLP and $94.2 \pm 1.4\%$ for RBFN. Finally, the RBFN reaches a state-of-the-art accuracy of 95.6% on a 350 feature dataset, which means it correctly predicts a gender of 19 out of 20 users.

5. CONCLUSION

Authors implemented a key-logger software that generates a comma separated file containing entries for both key-press and key-release events for each key pressed by a user. They received almost 250 csv log files from participants of different age, mostly between 26 and 45 years old with various education levels and average time spent using the computer daily. The samples are evenly distributed between genders. They built a dataset, computing for each log file the average duration of a key press and the average time between two consecutive presses for each pair of keys. To reduce time complexity of training on a dataset which initially had over 10,000 features, they sorted the features by the amount of information they carry using entropy and information gain. They experimented with 5 different machine learning models, training them on different sets of features, their sizes varying between 50 and 400, while also trying to estimate optimal parameter values for each of the models.

Results show that only a few percents of all key press durations and digram latencies are sufficient to train a robust classifier for gender prediction. In addition, most popular machine learning models yield similar accuracies for varying number of training features and a wide range of model parameters. Lastly, it is possible to train models that achieve state-of-the-art results in the field of gender recognition with keystroke dynamics.

In practice, this approach might pose some security issues, revealing passwords or sensitive personal data. To account for that, authors propose to do the feature generation on user's computer, thus avoid sending any personal data over the network as the features themselves give no information about actual text written by a user. Implementing such system within an operating system, that would forward the features to some remote server, which would process the data and only send final prediction to the communication partner or a forensic. Because the features are not straightforward, it would be hard for a malicious user to alter the result by changing the typing speed. As a possible improvement, authors propose enlarging the dataset and using it to also predict other characteristics, such as age.

6. REFERENCES

[1] M. M. Al-Jarrah. An anomaly detector for keystroke dynamics based on medians vector proximity. *Journal of Emerging Trends in Computing and Information Sciences*, 3(6):988–993, 2012.

[2] J. S. Alowibdi, U. A. Buy, and P. Yu. Language independent gender classification on twitter. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, ASONAM '13, pages 739–743, New York, NY, USA, 2013. ACM.

[3] B. D. Barkana and J. Zhou. A new pitch-range based feature set for a speaker's age and gender classification. *Applied Acoustics*, 98:52 – 61, 2015.

[4] N. Bartlow and B. Cukic. Evaluating the reliability of credential hardening through keystroke dynamics. In *2006 17th International Symposium on Software Reliability Engineering*, pages 117–126, Nov 2006.

[5] E. Eidinger, R. Enbar, and T. Hassner. Age and gender estimation of unfiltered faces. *IEEE Transactions on Information Forensics and Security*, 9(12):2170–2179, Dec 2014.

[6] R. Giot and C. Rosenberger. A new soft biometric approach for keystroke dynamics based on gender recognition. *International Journal of Information Technology and Management*, 11(1-2):35–49, 2012. PMID: 44062.

[7] T. R. Kalansuriya and A. T. Dharmaratne. Neural network based age and gender classification for facial images. *ICTer*, 7(2), 2014.

[8] X. Li, S. J. Maybank, S. Yan, D. Tao, and D. Xu. Gait components and their application to gender recognition. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(2):145–155, March 2008.

[9] J. V. Monaco, N. Bakelman, S. H. Cha, and C. C. Tappert. Developing a keystroke biometric system for continual authentication of computer users. In *2012 European Intelligence and Security Informatics Conference*, pages 210–216, Aug 2012.

[10] I. Tsimperidis, A. Arampatzis, and A. Karakos. Keystroke dynamics features for gender recognition. *Digital Investigation*, 24:4 – 10, 2018.

[11] I. Tsimperidis, V. Katos, and N. L. Clarke. Language-independent gender identification through keystroke analysis. *Inf. & Comput. Security*, 23:286–301, 2015.

Ocenjevanje časa snemanja zvočnih posnetkov

Borut Budna*
Fakulteta za računalništvo in informatiko
Večna pot 113
Ljubljana, Slovenija
bb6264@student.uni-lj.si

Robert Cvitkovič†
Fakulteta za računalništvo in informatiko
Večna pot 113
Ljubljana, Slovenija
rc5289@student.uni-lj.si

Timotej Gale‡
Fakulteta za računalništvo in informatiko
Večna pot 113
Ljubljana, Slovenija
tg4956@student.uni-lj.si

Povzetek

To delo obravnava problem ujemanja ENF (frekvenca električnega omrežja, ang. electrical network frequency) vzorcev v sklopu ocenjevanja časa snemanja zvočnih posnetkov. V delu je po navdihu vizualne primerljivosti predlagan in opisan nov kriterij podobnosti (bitna podobnost) za merjenje podobnosti med dvema ENF signaloma. Predstavitvi kriterija sledi opis iskalnega sistema, ki najde najboljše ujemanje za določen testni signal ENF v velikem obsegu iskanja na referenčnih ENF podatkih. Z empirično primerjavo z drugimi priljubljenimi kriteriji podobnosti je v delu dokazano, da je predlagana metoda bolj učinkovita od najsodobnejših tehnik. Na primer, v primerjavi z nedavno razvitim algoritmom DMA omenjena metoda doseže 86.86% nižjo relativno napako in je za približno 45-krat hitrejša od DMA. Na koncu je predstavljena strategija preizkusa edinstvenosti za pomoč človeškim ocenjevalcem pri zagotavljanju natančnih odločitev, posledično je metoda praktično uporabna v forenziki.

Ključne besede

Zvočni časovni žig, Frekvenca električnega omrežja, Prepoznava vzorcev, Zaporedna podobnost, Obsežno iskanje

1. UVOD

Frekvenca električnega napetostnega omrežja (ENF) je prepoznana kot edinstven prstni odtis, ki je nenamerno vgrajen v zvočne in video posnetke. Osredotočen na nominalni frekvenci 50 Hz (npr. Singapur) ali 60 Hz (npr. Združene države Amerike), ENF signal vsebuje naključna nihanja skozi čas s približkom nominalne vrednosti in se pojavi kot zaporedje nihajočih frekvenčnih vrednosti. Ta naključna nihanja so skladna na različnih mestih znotraj istega elektroenergetskega omrežja. Posledično imajo posnetki, ki so zajeti na različnih krajih hkrati, iste prstne odtise ENF, ki kažejo na enaka nihanja. Torej, če želimo preveriti, ali sta bila posnetka zajeta hkrati, je ena izmed možnih rešitev primerjava njunih ENF prstnih odtisov.

Digitalna snemalna naprava lahko zajema ENF signal iz lokalnega električnega omrežja, če je iz le-tega neposredno napajana ali je nameščena v bližini naprav, ki se polnijo. Natančneje, električni transformator, ki je neposredno priključen na napajanje, se lahko uporabi za zapisovanje in shranjevanje čistih ENF signalov tekom daljšega časovnega obdobja (referenčna baza podatkov). Za posnetke iz drugih naprav, kot so prenosni zvočni snemalniki in stacionarni nadzorni sistemi, se ENF signali primerjajo z referenčnimi signali. Najboljše vizualno ujemanje nakazuje na čas snemanja posnetkov. Ta aplikacija se imenuje ocena časa snemanja in je v multimedijski forenziki zelo uporabna.

Vizualna primerjava se uporablja samo za iskanje ujemanja v zelo kratkem referenčnem ENF zaporedju. Za veliko referenčno bazo podatkov je potrebna avtomatska primerjava in iskalna rutina za lokalizacijo najboljših ujemanj v referenčnih podatkih. Za poenostavitev interpretacije v tem delu imenujemo ENF referenčno bazo kot referenčni ENF in ENF vhodnega (preiskovanega) zvočnega posnetka kot testni ENF. Kot smo omenili, sta tako testni kot referenčni ENF predstavljena kot zaporedje nihajočih vrednosti. Pri nalogi ocenjevanja časa snemanja je referenčno ENF zaporedje običajno veliko daljše od testnega ENF. Klasični algoritmi iskanja so osnovani na minimalni povprečni kvadratni napaki (MMSE) in maksimalnem korelacijskem koeficientu (MCC). Delujejo tako, da primerjajo dani testni ENF z vsemi možnimi referenčnimi ENF segmenti enake dolžine. Minimum ali maksimum označuje najboljše ujemanje.

1.1 Pregled področja

Analiza frekvenc električnega napetostnega omrežja (ang. *ENF analysis*) je uveljavljena forenzična tehnika za preiskovanje zvočnih in video posnetkov, ki deluje na podlagi primerjav sprememb šumenja ozadja posnetka z zgodovinskimi posnetki nihanja frekvenc v električnem omrežju [5]. Nihanja v frekvenci so skladna v celotnem električnem omrežju, tudi ko so merjena na fizično oddaljenih lokacijah. Za potrebe tehnike je potreben zajem in hramba zgodovinskih posnetkov šuma električnega omrežja, te posnetke običajno hranijo forenzične institucije.

Podobno kot elektromagnetni valovi iz električnih povezav vplivajo na naprave za zajem zvoka in posledično prenašajo podatke o frekvenci omrežja, lahko iste podatke zajamemo iz video posnetkov, ki so bili zajeti blizu luči. Luči namreč svetijo v skladu z višino trenutne električne napetosti (ta je odvisna od frekvence napetosti), posledično lahko iz moči

osvetlitve izluščimo frekvenco omrežja. Pri tem obstaja nekaj omejitev: tehnika deluje zgolj z določenimi tipi luči, zaradi odstopanja hitrosti vzorčenja kamer od frekvence utripanja luči je problematičen tudi t.i. *aliasing*. [7]

Šum v električnem omrežju obravnavamo kot časovno odvisno digitalno oznako, ki omogoča [8]:

- ugotavljanje časa nastanka posnetka,
- potrjevanje ali izpodbijanje domnevnega časa nastanka posnetka,
- povezovanje/usklajevanje več posnetkov,
- odkrivanje sprememb posnetka.

V tem delu se osredotočimo na ocenjevanje časa snemanja zvočnih posnetkov in pripadajoče kriterije podobnosti, s katerimi v referenčni bazi poiščemo ujemanje s testnim posnetkom. Za potrebe zagotavljanja visoke natančnosti iskanja najprimernejšega ujemanja so se pretekla dela osredotočala na dva aspekta:

1. Ekstrakcija vzorca testnega ENF [4, 6, 3, 2, 9].
2. Iskalni algoritem, ki je neobčutljiv na šum [10].

Več pozornosti je bilo namenjeno prvemu aspektu, ki zajema procesiranje zvočnega signala, kot drugemu, ki zajema razpoznavanje vzorcev. Za zniževanje stopnje šuma v signalu in zviševanje točnosti ocene časa so bili razviti razni avtoregresivni modeli [6], evalvirani harmonični modeli [2, 9, 3] in uporabljeni filtri, ki delujejo na osnovi mediane [4, 10]. Za iskanje ujemanj v [10] avtorji predlagajo dinamični algoritem za ujemanje na osnovi praga DMA (ang. *threshold based dynamic matching algorithm*) za odpravo šuma znotraj frekvenčnega pasu in problema s frekvenčno resolucijo. DMA se odreže bolje kot tipični iskalni algoritem na osnovi MMSE. Algoritem DMA poveča robustnost zaznavanja vzorcev, a je računsko zahtevnejši. Posledično je njegova uporaba omejena na preverjanje časovnih značk zvoka, kjer je obseg referenčnega ENF majhen in določen s strani uporabnika [10]. Pri iskanju ujemanja testnega ENF v referenčni bazi je učinkovito iskanje enako pomembno kot točnost ujemanja [12].

Kljub uporabnosti tehnike ENF pa ima le-ta v splošnem nekaj slabosti in omejitev. Zanesljivost tehnike je močno pogojena s kvaliteto baze vzorcev, v kateri iščemo podobnosti. Vzorci v bazi imajo navadno visoko stopnjo t.i. samopodobnosti (ang. *self-similarity*). Možnost obstoja frekvenčnega odstopanja dodatno poviša možnosti napačne določitve časa posnetka. To je še posebej opazno pri krajših posnetkih (trajanje manj kot 10 minut) v povezavi s postopki, ki za ujemanje uporabljajo minimalno kvadratno napako. [11]

1.2 Prispevek dela

V tem delu je predlagana nova mera podobnosti za ocenjevanje razdalj med dvema ENF zaporedjema. Na podlagi te podobnosti se razvije sistem hitrega iskanja najboljšega ujemanja testnega ENF v dolgem referenčnem ENF zaporedju. Prispevki tega dela v primerjavi s prejšnjimi deli so:

- Zbiranje dveh vrst posnetkov (Singapur) in sicer posnetke moči, ki vsebujejo referenčne ENF in zvočnih posnetkov, ki vsebujejo testne ENF signale.
- Razvoj bitne podobnosti (bSim) za primerjavo dveh ENF zaporedij. Ideja za bSim temelji na človeškem kriteriju vizualne primerljivosti, ki neposredno meri delež ujemanja med dvema ENF zaporedjema. Eksperimentalni rezultati kažejo, da ima merilo bSim, še posebej pa njegov proces binarizacije, pomembno vlogo pri hitrem in natančnem iskanju ENF ujemanj.
- Gradnja iskalnega sistema, ki bistveno presega predhodne sisteme za ocenjevanje časa glede na natančnost in računsko učinkovitost.
- Uporabimo iskalno strategijo Top-n za pomoč človeškim preiskovalcem pri potrjevanju ocenjenih časov. Posledično je predlagana metoda praktično uporabna v forenziki.

2. OZNAČENI ENF SIGNALI (SINGAPUR)

Mesto Singapur je v celoti pokrito z enim elektroenergetskim omrežjem s katerim upravlja društvo SP PowerAssets, ureja pa ga vladna agencija (organ za energetske trge). Kot eno najbolj zanesljivih elektroenergetskih omrežij na svetu, ENF služi kot dober časovni žig v zvočnih posnetkih znotraj Singapurja. Zaradi vzpostavitve praktičnega sistema za oceno časa snemanja zvočnih posnetkov se vzpostavi prva podatkovna baza označenih ENF signalov v Singapurju (LESS). Podatkovna baza je sestavljena iz dveh podmnožic, ena je podmnožica referenčnih ENF posnetkov, zajetih v močnostih posnetkih, druga pa je podmnožica testnih ENF-jev, zajetih v zvočnih posnetkih.

2.1 Referenčni ENF iz močnostnega posnetka

Frekvenca napajalnega omrežja v Singapurju se ohranja na okoli 50 Hz +- 0.2 Hz. Čisti ENF podatki se lahko zajamejo s pomočjo digitalnih snemalnikov, ki so direktno priključeni na napajanje. Za snemanje močnostnih posnetkov (od 3. septembra 2013) se je uporabljala notranja zvočna kartica s frekvenco vzorčenja 400 Hz. Do zapisa članka so tako imeli avtorji na voljo več kot za 3 leta referenčnih ENF posnetkov. Za časovno frekvenčno analizo so nad vsakim posnetkom iz baze naredili hitro Furierjevo transformacijo in kvadratno interpolacijo z namenom ekstrakcije ENF signala. Poleg tega imajo vsi referenčni ENF posnetki samodejno označen čas snemanja.

2.2 Testni ENF iz zvočnega posnetka

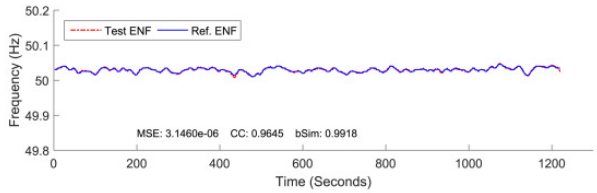
Postopek ekstrakcije ENF-ja iz zvočnega posnetka se nekoliko razlikuje od postopka z močnostnimi posnetki. Običajna frekvenca vzorčenja zvočnega posnetka je veliko večja, npr. za glasbo 44.1 kHz in 8kHz za govor. V ta namen moramo pred uporabo kratke Furierjeve transformacije za ekstrakcijo ENF-ja najprej poseči po metodah predproesiranja kot sta downsampling in pasovno odprto filtriranje. Konfiguracija in nastavitev parametrov so isti kakor v članku Hua-ja in ostalih.

Kvaliteta testnega ENF-ja je ponavadi nižja od referenčnih ENF-jev. To je zato, ker zvočni snemalniki niso nujno povezani z napajalnikom ampak so lahko polnjeni preko baterije.

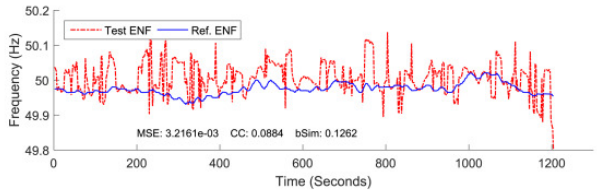
Prenosni zvočni snemalniki zajemajo zvok v razmeroma bolj hrupnem in bolj zapletenem okolju kakor snemalniki moči, zato čistost ENF signalov zvočnih posnetkov ni zagotovljena. Dejansko lahko prenosni snemalnik zajame veljaven ENF signal le če se nahaja v bližini opreme, ki se napaja direktno iz omrežja. V ta namen je bilo za snemanje izbrano mesto Singapur saj le-tega pokriva gosto elektroenergetsko omrežje in električna oprema (npr. gospodinjski aparati, ulične luči).

Od 30. junija 2016 do 24. avgusta 2016 so avtorji zbrali 187 zvočnih posnetkov na različnih območjih Singapurja. Ker so mobilni telefoni danes najpogostejša prenosna snemalna naprava so zvočne posnetke posneli z iPhonom (najpriljubljenejša naprava v letu 2016), Android telefonom in Windows telefonom. Kot pri posnetkih moči se časi zvočnih posnetkov beležijo avtomatsko s pomočjo mobilne aplikacije. Lokacije zajetih posnetkov so v skladu z dnevnimi aktivnimi območji lastnikov mobilnih telefonov. Na primer tri najbolj aktivna področja so pisarna, hrana in dom. Večina zvočnih posnetkov je dolgih med 20 in 40 minut.

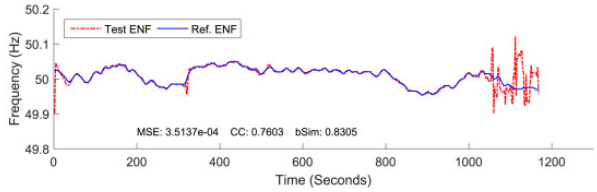
Z ročnim pregledom celotne testne množice, so avtorji ugotovili, da se popolno ujemanje ali popolna neusklajenost redko zgodi in da je delno ujemanje najpogostejše. Primer je prikazan na sliki 1 (c). Vidimo, da so glavni deli testne in referenčne ENF med seboj dobro prekrivajo, vendar neusklajenost nastopi v zadnjih 3 minutah.



(a) ENF signals (20 minutes) @ 2016-07-06 12:08:30



(b) ENF signals (20 minutes) @ 2016-08-02 20:30:13



(c) ENF signals (20 minutes) @ 2016-08-20 13:00:38

Slika 1: Vizualizacija ujemanja testne in referenčne ENF.

2.3 Parna podobnost med zaporedji

Vizualna primerjava je najbolj naraven način za ugotavljanje podobnosti dveh zaporedji. Ni pa učinkovito pri dolgih referenčnih ENF-jih in mora bit zato nadomeščena z avtomatsko primerjavo. V tem poglavju najprej pregledamo obstoječe mere podobnosti za zaporedja, nato pa predlagamo novo mero, ki bolje ustreza nalogi usklajevanja ENF.

2.3.1 *Vizualna primerjava*

Vizualna primerjava je najbolj časovno potraten a najnatančnejši način ocenjevanja podobnosti ENF. Za nalogo preverjanje časovne oznake, kjer je čas snemanja znan, se lahko hitro odločimo ali testni ENF ustreza referenčnim ENF. Vendar pa moramo, kadar ne poznamo časa snemanja, najdi podoben referenčni segment ENF v veliki bazi podatkov, kar je z vizualnim iskanjem nemogoče. Poleg tega vizualna primerjava ne poda numerične ocene podobnosti. Zaradi teh pomanjkljivosti se v praksi vizualna primerjava ne uporablja za ocenjevanje časa snemanja.

2.3.2 *Povprečje kvadrata napake*

Povprečna kvadratna napaka (mean squared error - MSE) je priljubljeno merilo za ocenjevanje podobnosti med dvema zaporedjema [10]. Za predstavitev testne ENF in referenčne ENF uporabljamo vektorja t in r. Mera MSE je podana z enačbo 1

$$MSE\,(t,r) = \frac{1}{N}\left\|t-r\right\|^2 = \frac{1}{N}\sum_{i=1}^{N}\left(t_i - r_i\right)^2. \qquad (1)$$

Vidimo lahko, da je vrednost vedno nenegativna in bližje kot je ničli, boljše je ujemanje.

2.3.3 *Korelacijski koeficient*

Alternativa meri MSE je Pearsonov korelacijski koeficient [10], ki je definiran kot

$$CC\,(t,r) = \frac{(t-\bar{t})\,(r-\bar{r})}{\left\|t-\bar{t}\right\|\left\|r-\bar{r}\right\|} = \frac{\sum_{i=1}^{N}\left(t_i-\bar{t}\right)\left(r_i-\bar{r}\right)}{\sqrt{\sum_{i=1}^{N}\left(t_i-\bar{t}\right)^2}\sqrt{\sum_{i=1}^{N}\left(r_i-\bar{r}\right)^2}} \qquad (2)$$

kjer je $\bar{t}$ aritmetična sredina zaporedja t. Vrednost CC leži v razponu [-1, 1], kjer vrednost 1 predstavlja natančno ujemanje. Baksteen (2015) [1] je dokazal, da je CC enak ničelno povprečnem MSE pod predpostavko, da je standardni odklon testnega ENF in referenčnega ENF enak. To predpostavko pa težko izpolnimo za potrebe iskanja v veliki referenčni bazi podatkov. Zato se MSE in CC uporabljajo samo ob določenih pogojih: CC deluje bolje kot MSE za zvočne posnetke krajše od 10 min ampak ima večjo časovno zahtevnost in zato ni primerno za iskanje na veliki količini podatkov.

2.3.4 *Bitna podobnost*

Zgornji dve meri združita lokalne razlike med posameznimi pari v dva vektorja. Na primer, kadar nastopajo veliki poskoki v testnem ENF to povzroči tudi veliko napako, kar posledično zelo vpliva na končen rezultat MSE. Drugače povedano MSE vključuje ne samo velikost ujemajočega se področja ampak tudi velikost neujemanja. To pa ni zaželeno

pri merjenju podobnosti ENF-ja. S tem, da imajo zaporedja ENF ničelno povprečje CC kriterij zmanjša vpliv, ki ga povzročajo velika lokalna neusklajenosti in je zato boljša mera podobnosti za robustne ENF-je. Kot že prej omenjeno pa izboljšavo dosežemo na račun večje časovne zahtevnosti. Po zgledu zgornjih mere še posebej vizualne primerjave avtorji predlagajo novo podobnost, imenovano bitna podobnost (bSim) za merjenje ENF podobnosti. Idejo lahko izrazimo z

$$bSim\,(t,r) = \frac{1}{N}\sum_{i=1}^{N} s_i, s_i = \begin{cases} 1, t_i \approx r_i \\ 0, t_i \not\approx r_i \end{cases}, \qquad (3)$$

kjer pogoj $t_i \approx r_i$ vrne 1, če se t_i ujema z r_i, drugače pa 0. Z binrizacijo lokalne razlike bSim poda delež ujemanja dveh ENF nizov. Kot človeški vizualni sistem, bSim ne računa točnih vrednosti razlike, ampak obravnava vso lokalna neusklajenost enako. Človeške oči zlahka ocenijo ujemanje, računalnik pa takšne odločitve ne more sprejet brez neke numerične mere. Podobnost $t_i \approx r_i$ je zato implementirana kot binarizirana funkcija $\|t_i - r_i\| < \theta$ in zgornjo enačbo prepišemo v

$$bSim\,(t,r) = \frac{1}{N}\sum_{i=1}^{N} s_i, s_i = \begin{cases} 1, \|t_i - r_i\| < \theta \\ 0, \|t_i - r_i\| \geq \theta \end{cases}, \qquad (4)$$

kjer je θ meja za ujemanje. Po binrizaciji postane razlika med dvema nizoma niz bitov, kjer enice prestavljajo področje ujemanja. Zato so avtorji mero poimenovali bitna podobnost. Podobno kot CC je rezultat bSima na omejenem področju [0, 1]. Vrednost predstavlja kolikšen delež testne ENF se ujema s referenčno ENF. Poleg tega je bSim časovno efektiven. V primerjavi s MSE kjer računamo kvadrat, bSim vzame isto vrednost ter njeno absolutno vrednost primerja z mejo θ. Obe meri pa sta manj zahtevni kot CC. Zaključimo lahko, da bSim izkorišča prednosti tako CC kot MSE, da doseže točno mero podobnosti v kratkem času.

3. OCENJEVANJE ČASA SNEMANJA

3.1 Obsežno iskanje

Brez podanega časa snemanja se preverjanje enakosti enako dolgih nizov ENF pretvori v iskanje najboljšega ujemanja testnega ENF v dolgem referenčnem ENF. Testni ENF moramo primerjat z vsakim segmentom iste dolžine v referenčnem ENF. Takšen postopek je znan kot poravnava zaporedja. Zanj uporabljamo matriko podobnosti ali matriko razdalj. Če predpostavimo, da sta dolžini testne ENF t in referenčne ENF r enaki N in M lahko izračunamo absolutno napako za vsak par ter dobimo matriko D velikosti NxM. Element d_{ij} je torej enak $d_{ij} = \|t_i - r_i\|$.

3.1.1 Binarizacija matrike podobnosti

Glede na bSim sta para elementa podobna, kadar je njuna absolutna napaka manjša od neke meje θ. Zato lahko matriko razdalje enostavno pretvorimo v matriko podobnosti s pomočjo $s_{ij} = (d_{ij} < \theta)$. Takšna matrika zaradi vsebovanih bitnih vrednosti namesto decimalnih zavzema manj prostora in je zato njena obdelava hitrejša. Med vsemi segmenti v referenčni ENF hočemo poiskati takšnega, ki ima maksimalno podobnost s testnim ENF. Če $r^{(k)}$ predstavlja segment dolžine N v referenčnem ENF potem je iskana funkcija oblike

$$\underset{k}{argmax}\; bSim\left(t, r^{(k)}\right) = \underset{k}{argmax} \sum_{i=i}^{N} \frac{s_{ij}}{N}, j = k+i-1, \quad (5)$$

kjer je s_{ij} (i,j) ti element matrike podobnosti in $argmax$ označuje maksimalen element. Parameter k je celo število med 1 in $M - N + 1$ in je s tem zagotavlja da so vsi referenčni segmenti dolžine N. Mesto največje podobnosti je tudi mesto kjer se testni in referenčni ENF najbolj ujemata.

3.1.2 Lokalizacija ujemajočega fragmenta

Vrednost bSim-a nam poda oceno, kako podobna sta odseka ENF, argmax pa točno lokacijo ujemanja. Vidimo lahko, da imamo pri testne ENF tri mesta ujemanja. Tako nastopi problem avtomatskega prepoznavanja teh manjši fragmentov. Namesto štetja enic predlagamo hitro rešitev s pomočjo XOR operacije. Algoritem dela tako, da najde spremembe zaporednih bitov (iz 1 v 0 ali obratno) ter kot rezultat poda indekse teh sprememb.

3.1.3 Ocenjevanje edinstvenosti vzorca ENF

Najpomembnejša predpostavka pri ocenjevanju časa posnetka je ta, da so vzorci ENF-ja edinstveni. Empirični preizkusi [10] so pokazali, da so vzorci ENF večinoma unikatni ampak so lahko v nekaterih primerih tudi zelo podobni. Zato je potrebno pred podajanjem ocene preveriti ali obstaja več mest ujemanja. Kratek zvočni posnetek ima večjo verjetnost ponavljanja kot dolg. Pretekle raziskave [10] so pokazale, da je za verodostojno oceno potreben posnetek dolžine vsaj 10 minut. Ker se vzorci ENF generirajo naključno bomo imeli v referenčni podatkovni bazi vedno podobnosti. Z vidika teorije verjetnosti je dolg vzorec ENF zaporedje nekaj kratkih vzorcev ENF, tako da se verjetnost podobnosti z drugim vzorcem manjša s večanjem dolžine. Iz tega lahko sklepamo, da za daljše meritve lažje napovemo čas snemanja.

3.1.4 Med-referenčna matrika podobnosti

Dolžina najdaljšega ponavljajočega se vzorca je odvisna od kriterija podobnosti in obsega iskanja reference. Za iskanje podobnosti uporabljamo enak postopek kot smo ga opisali v prejšnjem razdelku. S pomočjo matrike podobnosti ter v prejšnjem razdelku opisanega algoritma lahko tako najdemo najdaljše zaporedje enic. Odkrijemo, da v enournem referenčne ENF posnetku nastopa najdaljše ponavljajoče zaporedje dolžine 94 sekund. Iz tega sklepamo, da bodo imeli vsi krajši posnetki vsaj dve ujemanji v referenčnih podatkih. Takšno ponavljanje pa seveda ni zaželeno saj onemogoča točno napoved časa posnetka.

Kadar poskus ponovimo na daljšem referenčnem zaporedju ugotovimo, da je najdaljše ponavljajoče zaporedje tudi daljše od tega na manjših podatkih. Iz tega sklepamo, da večji kot bo obseg iskanja daljši bo najdaljši podvojen vzorec.

3.1.5 Meja podobnosti θ

Manjši kot je prag podobnosti krajši je najdaljši ponavljajoči se vzorec. Kot prej omenjeno meja θ predstavlja numerično predstavitev vizualnega kriterija ujemanja. Pri našem testiranju na enournih referenčnih podatkih lahko odstranimo ponavljanje vzorca z manjšim θ . V najboljšem primeru bi mejo postavil na nič a to v realnosti ni uporabno.

3.1.6 *Pregled edinstvenosti*

Iz zgornje analize lahko sklepamo, da za edinstvenost vzorca uporabimo manjši obseg iskanja ter ožji kriterij ujemanja a nobenega v realnosti ne moremo izpolniti. Ne moremo se zanesti na to, da uporabnik poda dobro oceno čas snemanja, da bi tako pravilno zmanjšali obseg iskanja. Prav tako ne obstaja točnega ujemanja med testnim in referenčni ENF-jem in zato prag ujemanja ne moremo postavit na nič. Namesto tega, da poskušamo zmanjšat ponavljanje vzorca raje uporabimo metodo Top-n (n>1) ter preverimo ali je prvi izmed dobljenih vzorcev edinstven. Ideja pri tem postopku je, da če je prvi veliko boljše ocenjen kot drugi to nakazuje na edinstvenost najdenega vzorca. Kadar sta najboljši dve oceni zelo podobni pa ne moremo sklepat na pravilnost napovedi. Takšna strategija je enostavna ampak pomembna za ocenjevanje časa snemanja. Njene prednosti so:

- Zaradi pregleda ni več potrebna predpostavka edinstvenosti vzorca ENF. Predlagan pregled prepozna podvajanje in to posebej označi. To nam omogoča, da ne napovemo čas za vzorec, ki ne vsebuje ENF.
- Ocenjen rezultat je neposredno odvisen od dolžina posnetka. Daljši posnetki imajo še vedno večjo verjetnost, da so edinstveni. To pa ne pomeni, da so dolgi posnetki vedno edinstveni ter kratki nikoli.
- Združuje avtomatično iskanje z numerično evalvacijo ter vizualno primerjavo najboljših ujemanj. Pri forenzičnem dokazu časa snemanja moram oceno podat strokovnjak. Tako namesto ene ocene dobimo Top-n primerjav, kar je bolj informativno.

4. EKSPERIMENTI IN ANALIZA

V prejšnjih razdelkih smo preučevali fenomen podvajanja vzorcev ENF v referenčnih podatkih in preverili edinstvenost za testne ENF posnetke. V tem poglavju eksperimentiramo z zvočnimi podatki v bazi LESS.

4.1 Postavitev eksperimenta

Za podan tesni ENF poiščemo optimalno ujemanje v referenčni bazi. Čas pri katerem se niza ujemata se vzame kot kandidat za napoved časa snemanja. Ko primerjamo dobljene čase z realnimi upoštevamo toleranco 1 minute, kar je v povezavi s tem, da ljudje navadno merimo čas na minuto natančno. Kot glaven način ocenjevanja uporabimo Top-n napako. Ta mera poda delež pravilnih napovedi, ki niso bili v Top-n. Za uporabo v forenzičnih preiskavah smo dodali tudi meri preciznost in priklic.

4.2 Rezultati

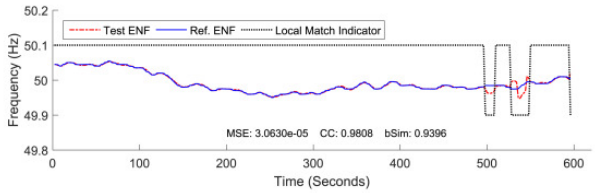
187 testnih ENF nizov iz LESS baze primerjamo s referenčnim nizom dolgim 240 dni. Dolg referenčni niz nam omogoča, da pokažemo sposobnost algoritma na velikih podatkih. Mejo podobnosti θ smo nastavili na vrednosti med 0,001 in 0,01. Najmanjša napaka pri Top-1 je 3,21% in je bila dosežena s optimizacije θ na intervalu [0,005; 0,007].

4.2.1 *Primerjava s sodobnimi metodami*

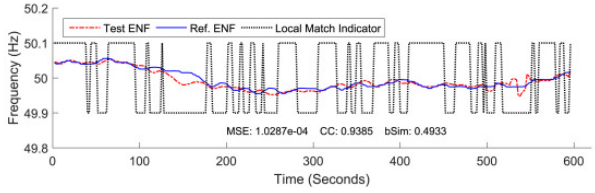
Predstavljeno metodo smo poimenovali maksimalen bSim in jo primerjali s sodobnimi metodami: dynamic matchin algorithem (DMA), minimalen MSE in maksimalen CC. Zaradi velike časovne zahtevnosti DMA smo iskanje omejili na ozek pas dolžine 26 ur. To vrednost smo izbrali ob predpostavki, da za posnetek poznamo dan nastanka ter temu dodamo po eno uro za prekrivanje s prejšnjim in naslednjim dnevom. Predlagana metoda je tako natančnejša kot hitrejša v primerjavi s ostalimi sodobnimi metodami. Maksimalen CC in minimalen MSE imata primerljive rezultate a je prvi nekoliko počasnejši. DMA zmanjša napako za kar 9,53% a se čas iskanja nekajkrat podaljša. Minimalni bSim je dosegel najmanjšo napako (86,86% manjšo kot DMA) in tudi najhitrejši čas. Ko primerjamo dobljene rezultate s temi, ki so bili izvedeni na daljših referenčnih podatkih opazimo, da imamo zaradi ožjega iskalnega okna manj ponavljan in je posledično natančnost tudi večja. Ugotovimo tudi, da se čas iskanja spreminja linearno glede na širino iskalnega okna.

4.3 Pomen preiskave edinstvenosti

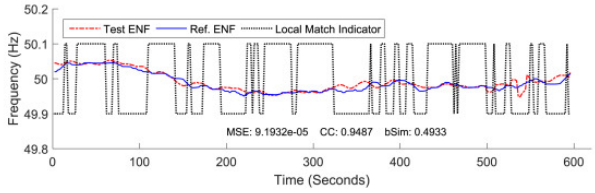
Vidimo, da je predlagana metoda učinkovita in natančna a kljub temu napaka ni enaka nič. *Precision* je zelo visok z 96,79% a ni enak 100%. Najbolje ocenjenemu rezultatu v povezavi z forenzično analizo zato ne moremo zaupat. Namesto enostavne izbire najboljše ocenjenega rezultata priporočamo izbito Top-n rezultatov.



(a) Reference ENF (10 minutes) @ 2016-08-23 23:20:54



(b) Reference ENF (10 minutes) @ 2016-04-13 23:49:05



(c) Reference ENF (10 minutes) @ 2016-04-22 18:46:24

Slika 2: Vizualizacija Top-3 rezultatov.

Na sliki 2 lahko vidimo primer takšnega Top-3. Ko primerjamo (a) s (b) in (c) lahko opazimo tri razlike:

1. Pri Top-1 sta testni in referenčni niz vizualno bolj podobna oziroma imata manj neujemanj.
2. Pri Top-1 je vrednost bSim veliko večja kot pri ostalih dveh.
3. Pri Top-1 imata testni in referenčni niz najdaljše zaporedno ujemanje.

Na podlagi teh izjav bi lahko ekspert sklepal, da je Top-1 pravilno izbrano področje ujemanja. Ko primerjamo tega z resničnim vidimo, da smo se zmotili zgolj za 2 sekundi. Zgoraj napisano lahko definiramo tudi formalno tako, da $bSim_1$, $bSim_2$ in $bSim_3$ predstavljajo vrednosti bSim za Top-3 rezultate. Po drugi točki mora bit rezultat Top-1 veliko večji od ostalih dveh. Tako definiramo pomembnostno odprtino (significance gap - sg):

$$sg = 2bSim_1 - bSim_2 - bSim_3. \quad (6)$$

Zanesljiv rezultat mora imet visoko vrednost sg drugače ga označimo kot nezanesljivega. Kljub temu, da je preiskava edinstvenosti izvedena s pomočjo eksperta nam z izbiro velikega sg metoda omogoča odstranit človečki pristranskost in nam poda samo zanesljive rešitve.

5. POVZETEK

V tem članku po navdihu vizualne primerjave predlagamo bSim (bitno podobnost) kot mero podobnosti ENF za namen ocenjevanja časa zapisa. Skozi eksperimentalno delo smo pokazali, da je bSim tako natančnejši kot hitrejši od klasičnima metodama MSE in CC. Predlagana metoda tudi presega najsodobnejši DMA algoritem z znatnim izboljšanjem, t.j. relativno napako zmanjša za 86,86% (z 20,32% na 2,67%) in za 45x pohiti odgovor iskanja (41,0444 s v primerjavi s 0,8973 s). Čeprav smo zagotovili veliko bolj natančno rešitev za nalogo časovne ocene, smo poudarili pomen človeškega preizkusa v forenzičnih aplikacijah in predlagala novo strategijo preverjanja edinstvenosti vzorcev ENF, tako da vizualno primerjamo Top-n rezultatov. Ta strategija podaja strokovnjakom podrobnosti prepoznavanje ujemanju vzorcev in pomaga pri filtriranju neuspehov pri zbiranju vzorcev ENF, t.j. zvočno snemanje morda ne bo zajelo veljavnega vzorca ENF lokalnega električnega omrežje. Eksperimentalno delo smo izvedli na lastno zbirko signalov ENF v Singapur, podatkovno zbirko LESS. Pokazali smo, da predlagani sistem omogoča obravnavo problem prilagajanja vzorcev ENF v kontekstu določitve ocene časa snemanja in v nadaljnjih prizadevanjih lahko analiziramo vpliv okolja pri zbiranju zvočnih posnetkov in nato izboljšamo kakovost zbranih testnih ENF.

LITERATURA

[1] T Baksteen. "The Electrical Network Frequency Criterion: Determining The Time And Location Of Digital Recordings". V: (2015).

[2] Dima Bykhovsky in Asaf Cohen. "Electrical network frequency (ENF) maximum-likelihood estimation via a multitone harmonic model". V: *IEEE Transactions on Information Forensics and Security* 8.5 (2013), str. 744–753.

[3] Jidong Chai in sod. "Source of ENF in battery-powered digital recordings". V: *Audio Engineering Society Convention 135*. Audio Engineering Society. 2013.

[4] Alan J Cooper. "An automated approach to the Electric Network Frequency (ENF) criterion: theory and practice." V: *International Journal of Speech, Language & the Law* 16.2 (2009).

[5] Alan J Cooper. "The electric network frequency (ENF) as an aid to authenticating forensic digital audio recordings–an automated approach". V: *Audio Engineering Society Conference: 33rd International Conference: Audio Forensics-Theory and Practice*. Audio Engineering Society. 2008.

[6] Ravi Garg, Avinash L Varna in Min Wu. "Modeling and analysis of electric network frequency signal for timestamp verification". V: *Information Forensics and Security (WIFS), 2012 IEEE International Workshop on*. IEEE. 2012, str. 67–72.

[7] Ravi Garg, Avinash L Varna in Min Wu. "Seeing ENF: natural time stamp for digital video via optical sensing and signal processing". V: *Proceedings of the 19th ACM international conference on Multimedia*. ACM. 2011, str. 23–32.

[8] Catalin Grigoras. "Digital audio recording analysis–the electric network frequency criterion". V: *International Journal of Speech Language and the Law* 12.1 (2005), str. 63–76.

[9] Adi Hajj-Ahmad, Ravi Garg in Min Wu. "Spectrum combining for ENF signal estimation". V: *IEEE Signal Processing Letters* 20.9 (2013), str. 885–888.

[10] Guang Hua, Jonathan Goh in Vrizlynn LL Thing. "A dynamic matching algorithm for audio timestamp identification using the ENF criterion". V: *IEEE Transactions on Information Forensics and Security* 9.7 (2014), str. 1045–1055.

[11] Maarten Huijbregtse in Zeno Geradts. "Using the ENF criterion for determining the time of recording of short digital audio recordings". V: *International Workshop on Computational Forensics*. Springer. 2009, str. 116–124.

[12] Alex Kantardjiev. *Determining the recording time of digital media by using the electric network frequency*. 2011.

Forensics of Programmable Logic Controllers

Jan Gulič
University of Ljubljana
Večna pot 113
Ljubljana, Slovenia
jg8698@student.uni-lj.si

Tilen Nedanovski
University of Ljubljana
Večna pot 113
Ljubljana, Slovenia
tn3626@student.uni-lj.si

Julija Petrič
University of Ljubljana
Večna pot 113
Ljubljana, Slovenia
jp8874@student.uni-lj.si

ABSTRACT

A programmable logic controller in many regards bears resemblance to a general-purpose computer or a microcontroller, but possesses important characteristics that make its significance in industrial automation much more prominent. This shows in the fact that most critical infrastructure today heavily relies on PLCs and other industrial control systems. Regardless of their value, little concern has been given to the security of said systems in the past. This is due to the fact that initially many devices used in an industrial automation, along with PLCs, were meant to be used in isolation – disconnected from other devices in the industrial environment. As industrial control systems evolved, they have started to rely heavily on the network and use of internet-based standards to share valuable data within large corporate networks. Hence, they have become vulnerable to a completely new set of exploits, that were traditionally used to target computers in a network. This has changed for the better over the years in which the industrial automation has become widespread.

In this work we give a primer on PLCs and their architecture, an overview of possible vulnerabilities and ways of intrusion and forensic challenges associated with them. Furthermore, we characterize a particular PLC and give insights into its intricacies and inner workings. Additionally, the proprietary GE-SRTP protocol is presented and evaluated as a means to obtain data from the device in forensic investigation.

Keywords

forensics, programmable logic controllers, industrial control systems

1. INTRODUCTION

Programmable Logic Controllers are, and have for years been an indispensable technology behind automation in the manufacturing and processing industries. They are ubiquitous in critical infrastructure as they are commonly used in controlling physical processes that pertain to power, water, gas, transport and other systems that are vital for the development and prosperity. Today their use is considered commonplace in these areas, and advancements made throughout the years immeasurably impacted industrial automation. Their widespread has given rise to a vast range of devices and a great number of manufacturers who produce them. Nevertheless, a fair share of PLCs are commonly programmed with the same set of tools and programming languages that have first been used in the wake of their conception.

Due to the potentially serious consequences of a fault or malfunction, PLCs are of immense strategical significance and are therefore often a target for computer oriented crimes and acts of terrorism. Newfangled PLCs have in large been equipped with various security mechanisms to allow only legitimate firmware to be uploaded. On the other hand, a privilege to alter behavioural logic of a PLC can typically be gained by anyone with a network or a physical access. Since the injection of exploitative code is, unlike in traditional IT setting, made trivial, this allows for a variety of hostile and intrusive software.

When performing a forensic investigation of an industrial control system it is beneficial to not only inspect the field devices involved, but to take a closer look at the overall digital control system. These typically include a central server authority which is decisive of task delegation and performs data acquisition. More often than not, an omniscient server will use commodity operating system which enables a forensic investigator to use a standard set of digital tools pertaining to the forensics of computers. On the contrary, devices closer to industrial machinery and final control elements, such as temperature sensors and control valves, commonly rely on proprietary hardware and system software. Thus, specialized forensic tools and techniques might be required to carry out the investigation. Since the use of industrial control system devices does not extend far beyond the industry, these tools can be very limited in functionality, that is, if they exist at all.

This paper comprises an overview of PLC vulnerabilities and ways of intrusion, a description of one particular series of PLCs and a study of a proprietary network protocol used in communication with the PLCs from the very series.

2. PRIMER

A programmable logic controller is a computer-like device used to control equipment most commonly found in an industrial setting. It plays an important role of connecting, monitoring and managing physical processes such as smelting, precipitation hardening, comminution and nuclear fission with the help of centrifuges. Typically many PLCs along with other devices work in cohesion in what is considered an industrial control system. From a functional standpoint an industrial control system can be divided into control centers and field sites. A run-of-the-mill control center serves as a human-machine interface to the field sites, but can often also act as an engineering workstation and as record-keep. The field sites include sensors and actuators along with PLCs as a means of controlling them. For instance, a gas pipeline along with a PLC which monitors and controls the gas pressure can be considered a field site. The PLC continuously obtains pressure values of the compressed gas in the pipe through various sensors in the pipeline. If the pressure exceeds a certain threshold, it invokes an actuator, in this case a valve, to release some gas which reduces the pressure in the pipe. A control center that possesses a capability of an engineering workstation can be used to configure and program PLCs in the field sites. This is commonly done through a PLC vendor-specific software. The logic that dictates how a PLC should administer a physical process is ordinarily written in one of the languages defined in the IEC 61131 standard. To build on the previous example, a PLC is programmed to maintain pressure in the pipeline between 0.2 MPa and 0.4 MPa. Based on readings from the sensor, if the pressure of the gas is more than 0.4 MPa, the PLC opens the valve to release some gas until the pressure is reduced.

There might be no continuous physical link between control centers and PLCs in the field sites, but when there is, these entities form a network and communicate with one another according to ICS specific protocols such as Modbus and PROFINET or general-purpose network protocols such as Ethernet. The data that passes from PLCs to the control centers is interpreted by the so-called Human Machine Interface (HMI) and presented to the human operator who, based on the insights from the information obtained, can take applicable actions. In this manner, operational decisions can be made efficiently whilst maintaining safety, should anything out of misreckon or mechanical issue happen. Since the decisions made, and more importantly the actions taken by the operator, are of the highest priority they can override the behaviour programmed into the PLCs. For example, the operator might decide to turn on the valve to release some gas even when the pressure is less than the programmed threshold value.

Generally a PLC includes nothing short of a CPU one would find in a desktop computer, input and output interfaces, a programming interface, possibly an interface to a communication network, power supply and two types of memory. In large, this means a volatile RAM and a nonvolatile EEPROM memory, of which the latter is used to store firmware or an operating system and the control logic program. Input and output devices attach to the PLC through the corresponding interfaces. These devices are, from the PLC standpoint, most commonly segregated into groups of discrete and analog modules. Input and output modules are said to be memory-mapped, which means that the data received from input devices and data to be sent to the output devices is retained in separate areas of RAM memory, rather than a separate memory entirely. Interfaces to the discrete or analog input devices convert the signals received to logic levels comprehensible to the processor. Likewise, the output module interface converts data received from the processor to signals capable of driving the attached discrete or analog output devices. Architectural diagram outlining said components is shown in figure 1.

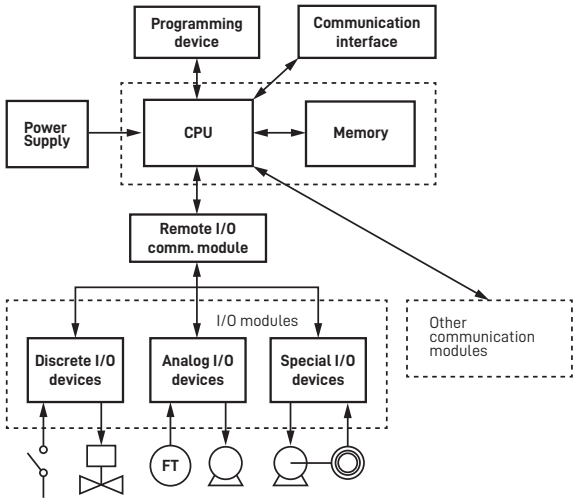


Figure 1: Conventional PLC architecture consisting of a CPU, memory and I/O modules, communication interface and other components.

While this might be reminiscent of a computer, some important characteristics distinguish a PLC from a general-purpose computer. Foremost, a PLC is designed to withstand pressures of industrial environment. This means it can be surrounded by otherwise detrimental impacts, such as substantial amount of electrical noise, vibration, extreme temperatures and humidity. PLCs were also conceived with reliability in mind as the Mean Time Between Failures (MTBF) metric is measured in years.

The PLC is often said to have a closed architecture. Contrary to an open architecture, found notably in computers and microcontrollers, a closed architecture is designed in a manner that makes it hard or impossible to add, upgrade or swap hardware components. This might mean no expansion of existing architecture is possible, or only proprietary hardware, which may require a license fee from the manufacturer, is compatible. This limits the capability of the architecture, but makes the design less complicated and therefore cost optimized.

PLCs and other industrial control systems in general were originally conceived on a premise that any entity operating within an ecosystem encompasses and executes legitimate firmware and software and strictly conforms to a protocol. It is because this assumption is still widely made today, that

many PLCs employ no security measure to prevent direct exploitation. PLCs do not verify the identity of other components with which they interact, they perform no data integrity checks on received message content, and carry out no encryption to preserve confidentiality of sensitive information [6].

The IEC 61131 was for many years, and in a limited way still is, the standard upon which many PLCs have been built. It was first published in 1992 and is considered the most prominent in industrial automation as it helped shape the way PLCs and other automation systems are produced, interacted with and dispatched. According to the IEC 61131-3 [4], several programming languages to impose logic and determine behaviour can be used. Structured Text (ST), Ladder Diagram (LAD), Instruction List (IL), Sequential Function Chart (SFC) and Function Block Diagram (FBD) can be purposed individually or in unison to fashion instructions for the control system to interpret and execute. Manufacturers implement these, otherwise formal languages, to different extents, thus cross compatibility is seldom ensured. Malware written in one of the IEC 61131-3-compatible languages consists of logic that purposefully alters the normal behaviour of a PLC. Abnormal behaviour in this case refers to the tampered sensor readings reported by the PLC, actions performed in stealth, intentional halt and other irregularities.

3. CYBERATTACKS ON PLC

Aggression against a PLC can happen on both the network and device levels. Malicious deeds that burgeon in the network include reconnaissance, man-in-the-middle (MITM) and denial-of-service (DoS) attacks.

Reconnaissance is not performed with harmful intent per se, but can be the preliminary step of gathering information in attack planning. The purpose of reconnaissance is to identify the make and the model of a PLC, the firmware it is installed with, the functionality it supports, etc. This information can be obtained either passively by eavesdropping or actively querying the PLC if the adversary has sufficient privilege to do so. Eavesdropping is in its passiveness hard to detect and investigate since it rarely leaves any traces in the network. This is untrue for active reconnaissance. In a general case, the adversary first looks for assigned PLC addresses by sending commonly understood request messages over the network. If a response is received for an address, this signifies that the address is used by a PLC. However, this approach leaves a significant number of undelivered messages and messages containing unknown PLC addresses in the network traffic log, which can be helpful in a forensic investigation. In what follows, the attacker might use a similar technique to discover what functionality is supported by the PLC at a specific address. If the offense exploits a vulnerability pertaining to a specific PLC make, the aim of reconnaissance will be to gather information on the make, model and the firmware version to identify the PLCs eligible for the attack. This manner of obtaining information will leave a lot of forensic artifacts in the network. Inspecting network traffic in time of the attack might spotlight the irregularities in normal operation, such as high message count with failed delivery, excessive amount of exception response messages from PLCs and the like. To stealth the aggression, the adversary will usually only explore a small number of addresses and only obtain information about a subset of functionalities that are essential to launch an offense.

The term man in the middle refers quite literally to the adversary positioned between the control center and the PLCs in the field sites. A mediator, the adversary can eavesdrop on the conversation between the two components of the industrial control system, by allowing all the traffic to pass through either way. At will, the attacker can choose to modify and manipulate the messages in the communication channel. The reasoning behind this is to fabricate a message, falsely assessed on its origin, that instructs a PLC to take actions out of ordinary or modify the data being reported by the PLC to the control center. While the former allows for interference with the device, the latter provides a measure of stealth that can keep the target of the offense in the dark. One approach to this technique is the so-called Address Resolution Protocol (ARP) spoofing. To establish its position in the network, the adversary modifies ARP tables, purposed in the resolution of IP and MAC addresses on the PLC as well as the computer running the HMI software, by associating the IP addresses in use by PLC and the HMI to the MAC address of the machine used by the adversary. Has this been achieved, the PLC and the HMI computer, instead of each other, now address the attacker's machine in messages they exchange throughout their communication, unknowingly redirecting the messages to a third party. Poisoning of ARP tables can be identified by analyzing the network traffic log for inconsistencies in IP-MAC associations, assuming the forensic investigator has the correct associations in the first place.

The intent of a Denial of Service (DoS) attack is to interfere with regular operation of a PLC and restrain it from executing logic and communicating with other devices in the industrial control system. Most commonly, a DoS attack is carried out by flooding the network with an excessive amount of packets, which exhausts the communication channel bandwidth. Thus a connection between the PLC and the control center is disrupted and the PLC is made unavailable to the rest of the system. The superfluous packets sent through the network might originate from one or many different sources, making it arduous to identify the adversary. Again, inspecting the network traffic log might give some insight to the forensic investigator. Another way to disengage and confine the PLC with a DoS is to directly exploit one of its vulnerabilities or a normal function in a malicious manner, such as a malformed packet causing the PLC to crash, in order to render the device inoperable.

Aggression that does not necessarily germinate in the network, but rather targets the PLC device directly includes attacks such as command injection and memory corruption. With modification of existing code and addition of unwanted logic into a PLC can grant an adversary an unauthorized control over the device. The injected code can modify the intended behaviour in all sorts of ways, such as making the PLC transition the state of a physical process to an abnormal one and changing the configuration of the PLC entirely, ultimately causing harm to the industrial equipment. Memory corruption on the other hand goes beyond control logic injection, and is able to alter parts of memory, not only con-

taining program code, but other data as well. An adversary can use the leverage to infect PLC's firmware and causes for instance, the input and output data to be changed arbitrarily.

3.1 Stuxnet

A particularly malicious software targeting industrial control systems, that serves well as an example of vulnerabilities of PLCs and other equipment in industrial automation, is Stuxnet. Prior to the discovery of the Stuxnet computer worm in June 2010, logic based PLC malware and prevention thereof received little attention. Stuxnet is a largely complex and lethal piece of malware that targets industrial control systems. It is certainly not the only one, but is the first one of its kind. A great amount of exploits that constitute Stuxnet was amassed by its creators in order to improve their chances of successful intrusion. This includes zero-day exploits, a Windows rootkit, the first ever PLC rootkit, antivirus evasion utilities, process injection and hooking code, network infection routines, peer-to-peer upgrades and a command and control interface [3]. The ultimate goal of Stuxnet was likely to sabotage a power plant or gas pipeline facility in Iran by reprogramming programmable logic controllers to operate outside of their specified boundaries. To infect the intended target, Stuxnet would be introduced into the target environment through a willing or an unaware third party. This could be a contractor or any person who had access to the facility. Once Stuxnet gained access to one of the computers within the facility it began to spread in search of programming devices, which are used to program and directly manage PLCs. Commonly these are Windows computers disconnected from the rest of the devices in the facility's network. For that reason Stuxnet employed LAN vulnerability to be able to propagate itself over the network and infected removable drives to be able to bridge the gap between the computers in the network and computers used in managing the PLCs.

4. SCADA

Before delving any further into the forensics of industrial control systems, the concept of SCADA system must be explicit to the reader. Supervisory control and data acquisition (SCADA) is a name used in referral to a system architecture that joins computers used in high-level supervisory management and PLCs or other peripheral devices used in immediate or direct management of the machinery into a single coherent control system. While the computers, which the operator can take advantage of to issue commands and monitor the system, take the supervisory role in the system, the PLCs and other discrete controllers directly connect to the field sensors and actuators and perform the necessary control logic based on the inputs received. SCADA systems bear a functional similarity to the distributed control systems (DCS), but use multiple means of interfacing even the geographically more dispersed industrial equipment (appliances). The centralized data acquisition and distributed control is crucial to the system operation. As SCADA systems evolved, they have started to rely heavily on the network and use of internet-based standards to share valuable data within large corporate networks. With evolution, however, came the ever-increasing risk of deliberate actions of malicious nature that can alter, disrupt, deceive, degrade and destroy the system or the information resident or transiting them.

In large, the cyberattacks targeting these systems exploit a vulnerability, which is an aspect or a defect of a system that can be used to compromise its intended functionality. This might be done remotely, from a distant location, or in close proximity, where the adversary has physical control over the system or parts of it. Following the exploit, several actions may be performed depending on the intentions behind the attack. Attacks on SCADA systems can be categorized into three groups, according to the type of exploit and the devices of which vulnerabilities are taken advantage of. Communication attacks occur in the network and include exploits such as SYN flooding and packet replay. Hardware attacks occur when an unauthenticated access is gained into the device and the system is set up for failure. Lastly, buffer overflow and SQL injections are types of software attacks.

5. RELATED WORK

The problem of detecting attacks in wireless sensor networks of SCADA systems is investigated in [1]. The authors have, based on analytical studies performed, fabricated a detailed classification of external attacks on sensor networks and came forth with a report on the impact the attacks have on various components of SCADA systems with respect to the contrived classes of attacks. They have compiled a review of different methods used in detection of wireless system intrusions, and have made significant the role of human in internal security threats. Findings from their study lead to the conclusion, that the most dangerous threats to information security are of anthropogenic nature and include unintentional personnel actions that establish favorable conditions for external attacks, the lack of qualification and competence of personnel in the field of information technology and security and a disregard of the requirements needed to bring about better security of information technology.

Research done in [6] focuses on the shortcomings of the network anomaly detection based on metadata, which includes message sizes, timing, command sequence and such, and on the state values of the physical process. Attacks against SCADA systems done by the authors and proven undetected, show the deficiency in metadata-based anomaly detection. The stealth was achieved by first hijacking the communication channels between the HMI and PLC devices, and then misleading the human operator involved by presenting fictitious information about the industrial process. The deception instills the operator into taking manual action. Solution to these man-in-the-middle attacks, as conceived by the authors, would be to secure the communication channel via cryptographic means. Nonetheless, the anomaly detection presented in the paper has been proven valuable as the attackers of such systems are restricted to only very deceptive attacks.

Authors of (the work) [7] suggest a probability risk identification based intrusion detection system (PRI-IDS) technique, which is based on network traffic analysis. This would be well employed in identifying replay attacks, which are a type of network attacks in which a transmission is repeated or delayed out of a malicious or fraudulent intent. Such attacks can be easily perpetrated on an unauthenticated and

unencrypted communication channel. The proposed technique has been shown to be robust and efficient at recognizing them.

Another intrusion detection technique has also been proposed in [5]. The initiative behind this paper is the fact that while privacy preservation techniques have become effective in in protecting sensitive information and detect malicious activities, they lack error detection and still disclose some amount of data considered sensitive or private. Authors propose a new privacy preservation intrusion detection technique which has been established on the basis of correlation coefficient and expectation maximisation clustering mechanisms. Used in unison, said mathematical tools are helpful in selecting important portions of data and recognizing intrusive events. Experimental results show that the technique is reliable and effective in recognizing suspicious activity and can be utilized in current SCADA systems.

To raise awareness about vulnerabilities of industrial control systems and propose security mechanisms to diminish the risk of a breach was the intent behind the paper [8]. In the preliminary part, the authors show how security procedures in existing standard protocols can be circumvented with little effort. In what follows, a suite of security protocols specifically designed for SCADA and DCS system is introduced. The proposed protocol incorporates point-to-point secure channels, authenticated broadcast channels and lastly the authenticated emergency channels and a revision thereof.

6. GE FANUC SERIES 90-30

To better envision what constitutes a vulnerability in a PLC, it is beneficial to get acquainted with the inner workings of a PLC. For this purpose the Fanuc Series 90-30 manufactured by General Electric is examined in this section.

For the most part, a typical PLC consists of several registers which can be read and written using a so called Human Machine Interface (HMI) which is typically a piece of computer software. General purpose data stored or read by the program, like other types of information, resides in the dedicated part of memory. The register memory, as it is referred to, dynamically holds data as dictated by the program. Other types of data are stored in separate non-overlapping areas of memory which may be, along with the register memory, referenced by different prefix types. The memory prefix types convention is used frequently and is necessary to understand the written logic.

The prefix `%R` is used when referring to the general purpose system registers. Prefixes `%AI` and `%AQ` address analog input and output data from a field device respectively. Discrete input and output memory sections are referred to by the prefixes `%I` and `%Q`. These sections contain data of all input modules received during the input and output scans. Prefix `%T` stands for temporary references. As the name suggests, the data referenced by this prefix is temporary, which means it may be lost during power failure or the transition between the run and stop modes of the PLC. Internal data is stored in the discrete momentary memory and is references by the prefix `%M`. Part of memory called system memory contains system status information, such as access timers, scan information, fault information pertaining to the PLC. Prefix `%S` is used to reference the data in this section, and prefixes `%SA`, `%SB` and `%SC` may be used individually to reference subregions within this part of memory. Lastly, with prefix `%G` it is possible to reference global data in the discrete global memory. Information on contact and coil status inhabits this region of memory.

6.1 HMI

PLCs in the GE Fanuc Series 90-30 line of devices can be controlled on the application layer via the DDE protocol. Operating this high on the abstraction layer stack means several pieces of software (in this case Excel or Wonderware) mediate between the programmer and the PLC. This means limited capability in terms of HMI, as well as performance penalties due to the indirect way communication is carried out.

A more direct approach, but from a technical standpoint more complex, is to use a communication protocol on the network layer. This allows for greater flexibility and control. A protocol developed by General Electric called GE-SRTP serves this purpose well. However, the GE-SRTP is a proprietary protocol which means little to no documentation is available. Fortunately enough, G. Denton et al. [2] have successfully reverse engineered the protocol and documented the process and the results in their preliminary work. As per [2], the process of reverse engineering was conducted in an technologically secluded environment. It consisted of the following components:

- GE Fanuc Series 90-30 (5-Slot Base IC693CHS397C, CPU 331, 120/ 240VAC Power Supply IC693PWR321X (includes Serial port), and CMM321 Ethernet Interface).
- Netgear Prosafe 16 port 10/100 Switch including Category 5 cables.
- HP Compaq NC6400 laptop.
- Software: MS Excel, Proficy Machine Edition 6.0, Wonderware Intouch v9.5, Wireshark, Wonderware IO servers for Host Communications version 8.1.101.0.

In the process of reverse engineering the authors employed Wireshark to capture all the network traffic between the HMI computer and the PLC device. According to the paper ladder logic for controlling a simple process was written for this purpose, and the code produced was uploaded to the PLC. Existing interfacing software operating on the application layer was used to interact with the PLC. With the help of this software, several requests were sent out and were, along with the responses given by the PLC, captured in the network. In order to gain comprehensive knowledge about the GE-SRTP protocol, the captured packets were analyzed, bit by bit for each outbound request sent to the PLC.

7. GE-SRTP PROTOCOL

The predecessor to the GE-SRTP protocol is the Serial Network Protocol (SNP). It serves a similar purpose as its

Table 1: Request message structure

Byte offset	Field type	Common value
1	type	0x02
2	unknown/reserved	0x00
3	sequence number	
4	unknown/reserved	0x00
5-8	text length	0x00
9	unknown/reserved	0x00
10-16	unknown/reserved	0x01
17	unknown/reserved	0x00
18-25	unknown/reserved	0x01
26	unknown/reserved	0x00
27	time (seconds)	0x00
28	time (minutes)	0x00
29	time (hours)	0x00
30	unknown/reserved	0x00
31	sequence number	
32-35	message type	0xc0
36-39	mailbox source	0x00 00 00 00
40	mailbox destination	0x10 0e 00 00
41	packet number	0x01
42	total packet number	0x01
43-47	service request code	
48-55	request type dependent	0x00

Table 2: Service request codes

Hex value	Service request code
0x00	PLC short status request
0x03	return control program names
0x04	read system memory
0x05	read task memory
0x06	read program memory
0x07	write system memory
0x08	write task memory
0x09	write program block memory
0x20	programmer logon
0x21	change PLC CPU Privilege Level
0x22	set control ID(CPU ID)
0x23	set PLC (run vs stop)
0x24	set PLC time/date
0x25	return PLC time/date
0x38	return fault table
0x39	clear fault table
0x3f	program store (upload from PLC)
0x40	program load (download to PLC)
0x43	return controller type and id information
0x44	toggle force system memory

successor, but was discontinued in favor of the newer protocol. As per [2] the differences between the protocols are hardly any. Between the two, the fields of information comprising the message often only differ in the byte offsets. When interfacing the PLC device through a GE-SRTP protocol, request packets are sent out to the network for each command specified by the programmer, and response packets carrying requested information are returned by the PLC in exchange. Both request and response packets are composed of a series of bits in a manner that conforms to the format specified by the GE-SRTP protocol.

7.1 Request

An overview of the request packet is shown in table 1. The payload of the request is composed of 55 bytes altogether. The `type` field denotes the type of the packet and is present in the payload of both request and response packets. To differentiate between the two, values `0x02` and `0x03` are used to denote the request and response respectfully. The `sequence number` field appears twice in the payload structure – byte offsets 2 and 30. Jointly, these fields are used in identification of request and response message pairs. On request, the HMI – or the master in general case – is responsible for generating a unique packet number which does not coincide with any other packet numbers currently in transmission. On response, the slave (in this case the PLC) should, according to the GE-SRTP protocol, copy this sequence number to the latter `sequence number` field in order to acknowledge the received request. Byte `43` enumerates the service request code, which varies according to the type of memory that is being referenced in the request. Of these values, the most important might be the `0x06` and `0x09` used to denote reading of and writing to the program memory requests respectfully. All other values along with the two are displayed in the table 2. The 5 bytes that follow are used to access different sectors of memory as described in the previous section. First of the 5 bytes denotes the segment and the amount of data to be accessed. Depending on the type, some sections of memory may be addressed in bits or bytes, while other are exclusively addressable in words. Word-addressable are the register and analog input and output memory, and among bit- or byte-addressable are discrete memory types, such as the discrete input and output memory and the discrete momentary memory. Complete reference of types of memory access with respect to the types of memory is shown in 3. The amount of data and the offset at which the data in memory will be accessed is given in two sets of two successive bytes. Bytes 44 and 45 hold the memory offset and bytes 46 and 47 specify the data length. In either case the order of the bytes in the two pairs is little endian.

7.2 Response

The response is structurally and in terms of content similar to the request. The `sequence number` is again at the offset 2 and serves the purpose of identifying the request–response pairs. Bytes `42` and `43` are used in responding with an error. Along with the `message type` field at offset 31, these bytes are used to report insufficient privilege level, a full PLC service request queue, illegal service request etc. If the PLC should respond with data, it must be stored in the 6 bytes following the first byte at offset `44`. The length of this section of the payload may vary in size as it accommodates more than 6 bytes of data. Lastly, the 5 bytes at the very end hold the information about the state of the PLC. Byte `50` indicates whether or not the master is authenticated into the program task. Value `0x00` denotes the fact that the master is logged into the task, and the value `0xFF` denotes the opposite. The privilege level at which the PLC operates is given by the `current privilege level` byte at offset 51. Two bytes that follow, convey information about the time it took to execute a program task most recently. This is referred to as the last sweep time. Constituent bits of bytes `54` and `55` report a status or a fault of different components of the PLC individually. For example, bit 2 reports if the

Table 3: Segment selection codes

Memory type	Bit-selector	Byte-selector	Word-selector
Discrete Inputs (%I)	0x46	0x10	
Discrete Outputs (%Q)	0x48	0x12	
Discrete Internals (%M)	0x4c	0x16	
Discrete Temporaries (%T)	0x4a	0x14	
%SA Discrete	0x4e	0x18	
%SB Discrete	0x50	0x1a	
%SC Discrete	0x52	0x1c	
%S Discrete	0x54	0x1e	
Genius Global Data (%G)	0x56	0x38	
Analog Inputs (%AI)			0x0a
Analog Outputs (%AQ)			0x0c
Registers (%R)			0x08

Table 4: Response message structure

Byte offset	Field type	Common value
0	type	0x03
1	unknown/reserved	0x00
2	sequence number	
3	unknown/reserved	0x00
4	text length	0x00
5-16	unknown/reserved	0x00
17	unknown/reserved	0x01
18-25	unknown/reserved	0x00
26	time (seconds)	
27	time (minutes)	
28	time (hours)	
29	unknown/reserved	0x00
30	unknown/reserved	value varies
31	message type	0xd4
32-35	mailbox source	0x10 0e 00 00
36-39	mailbox destination	0x20 5a 00 00
40	packet number	0x01
41	total packet number	0x01
42	status code	
43	minor status code	
44-49	return data	
50-55	PLC status	

I/O fault table has changed since it was last read.

7.3 Tool

Beyond the purpose of the paper [2], the knowledge gained was condensed into a tool that is capable of communication with the PLC over the network. This tools allows for direct connection without intermediary software or hardware – apart from the network hub. The intent behind the reason the tool was built is to read data from PLC-internal memory and identify possible attacks and exploits. From a forensic standpoint this is sufficient. Should a need to write and/or modify data in memory arise however, the tool is easily extended beyond its basic capabilities.

The tool has the following features [2]:

1. Reading the name of the program task currently running on the PLC.
2. Reading and writing values of all registers on the PLC device.
3. Reading PLC fault tables, I/O fault tables and CPU controller ID. The fault tables log all PLC and I/O modules abnormal operations such as low battery in PLC CPU or constant sweep exceeded.
4. Master logging into and out of the program task.
5. Changing the non-password protected privilege level of the master prior to a PLC service request.
6. Enabling/disabling I/O modules operation. I/O modules are used by the PLC to interface with a field devices or instruments. They are inserted in the PLC backplane slots and wired to instruments using manufacturer wiring diagram. We did not use an I/O module in our experiment because the headgate position movements were simulated using scripts built into the Wonderware application. If we had a 'real' headgate for experimenting, a position transmitter (instrument) would be physically connected to the gate. Once the gate starts to move either upward or downward, a proportional 4e20 milliamp (ma) signal would be transmitted on wires to an analog input I/O module. The signal would then be scanned by an analog input register memory (%AI03) in the PLC and then converted

(ma to feet) to engineering units for displaying the gate position on the HMI and/or in the PLC program task.

7. Changing the PLC state (RUN/STOP).

8. DISCUSSION

In the emergence of industrial automation and throughout its infancy, little to no emphasis was put on how to detect, prevent or keep at bay the malignant interactions with the industrial system, should any occur. This has changed for the better, due to the widespread use and the significance these industrial control systems bear by running critical infrastructure (such as water, air, gas ...).

Related work section contains many works of authors who strive towards better security and impermeability of industrial control systems. As it has been reported by several related works, and has also been shown by our research, PLCs and industrial control systems in general are becoming increasingly more attractive for adversaries. Exempting contemporary and state-of-the-art devices, it has been shown that gaining access to a PLC is rather straightforward. With key properties of the system configuration in mind, it is possible to start and stop the ladder logic execution, download and upload software, and send arbitrary commands to the PLC, without raising suspicion. Throughout the papers mentioned, various techniques to improve security of the industrial control systems have been proposed. Authors of [2] suggest better upkeep of existing devices, which includes activation of privilege levels, and careful planning and evaluation in designing protocols and firmware for devices to be released to the market. Should a breach happen however, reliable and featureful tools for forensic analysis are desired. Research done in [1] suggests that human error and maliciousness greatly impacts security. Often overlooked, these factors must not be underestimated. Intrusion detection techniques are proposed in [7, 5]. In [7] authors suggest network traffic analysis as a preliminary step to intrusion detection. Thus, out of ordinary activity can be detected and onslaughts like replay attack recognized. The idea behind intrusion detection technique in [5] is clustering and categorization of data considered private to selectively protect sensitive information and identify intrusive events when such information is queried from an unauthorized source.

9. CONCLUSIONS

Information given suffices for a fundamental understanding of the role PLCs play in industrial control systems, how they interact with other devices, and what challenges is a forensic investigator faced with when performing an analysis of said devices. Introduced have also been basic mechanisms by which various incursions into industrial control systems can be detected, analyzed and ultimately countermeasured. The integral part of this paper however, is an illustration of how only some ingenuity and knowledge about the PLCs is required to fully obtain the control of a, now outdated, device. Aside from that, the importance of freely available tools that are of great value in forensic investigation, is shown.

The consequence of transition from industrial control systems being deployed in isolation to large inhomogeneous networks is the increased probability of IT vulnerabilities giving way to directly control and manipulate devices constituting the system. The field of computer forensics is abundant with techniques and methods to investigate devices widespread in domestic as well as industrial use. However, PLCs and other industrial control systems have only gained traction in the forensic community in recent years, due to the increased amounts of malware released and vulnerabilities exposed. This shows in lack of proper forensic tools to analyze them, constraints posed on their resources and the proprietary nature of most of the hardware and software used to program and maintain them.

10. REFERENCES

[1] P. V. Botvinkin, V. A. Kamaev, I. S. Nefedova, A. G. Finogeev, and E. A. Finogeev. Analysis, classification and detection methods of attacks via wireless sensor networks in SCADA systems. *CoRR*, abs/1412.2387, 2014.

[2] G. Denton, F. Karpisek, F. Breitinger, and I. Baggili. Leveraging the srtp protocol for over-the-network memory acquisition of a ge fanuc series 90-30. *Digital Investigation*, 22:S26 – S38, 2017.

[3] N. Falliere, L. O. Murchu, and E. Chien. W32. stuxnet dossier. *White paper, Symantec Corp., Security Response*, 5(6):29, 2011.

[4] Programmable controllers – part 3: Programming languages. Standard, International Electrotechnical Commission, 2003.

[5] M. Keshk, N. Moustafa, E. Sitnikova, and G. Creech. Privacy preservation intrusion detection technique for SCADA systems. *CoRR*, abs/1711.02828, 2017.

[6] A. Kleinmann, O. Amichay, A. Wool, D. Tenenbaum, O. Bar, and L. Lev. Stealthy deception attacks against SCADA systems. *CoRR*, abs/1706.09303, 2017.

[7] T. Marsden, N. Moustafa, E. Sitnikova, and G. Creech. Probability risk identification based intrusion detection system for SCADA systems. *CoRR*, abs/1711.02826, 2017.

[8] Y. Wang. sscada: Securing SCADA infrastructure communications. *CoRR*, abs/1207.5434, 2012.

Pristopi digitalne forenzike za ekosisteme Amazon Alexa
Povzetek članka Digital forensic approaches for Amazon Alexa ecosystem [3]

Jan Blatnik
Fakulteta za računalništvo in informatiko
jb4634@student.uni-lj.si

Nejc Smolej
Fakulteta za računalništvo in informatiko
ns1024@student.uni-lj.si

Povzetek

Pametni zvočnik Amazon Echo je zelo pomemben za inteligentno virtualno pomočnico Alexo, ki je bila razvita s strani Amazon Lab126. Amazon Echo posreduje glasovne ukaze do Alexe, katera se sporazumeva z obilico združljivih naprav interneta stvari (angl. Internet-of-Things - IoT) in aplikacij drugih razvijalcev. IoT naprave, kot je Amazon Echo, ki so stalno vklopljene in povsod navzoče, so lahko zelo dober vir potencialnih digitalnih dokazov. Za podporo digitalnim preiskavam je pomembno razumevanje kompleksnega oblačnega ekosistema, ki omogoča uporabo Alexe. V članku so obravnavane metode digitalne forenzike, ki se nanašajo na ekosistem Amazon Alexa. Glavni del članka je namenjen novemu pristopu digitalne preiskave, ki združuje cloud-native forenziko in forenziko na strani odjemalca. Predstavljeno je tudi orodje CIFT (Cloud-based IoT Forensic Toolkit), ki omogoča identifikacijo, pridobitev in analizo tako cloud-native artefaktov v oblaku, kot odjemalsko-centričnih podatkov na lokalnih napravah.

1 Uvod

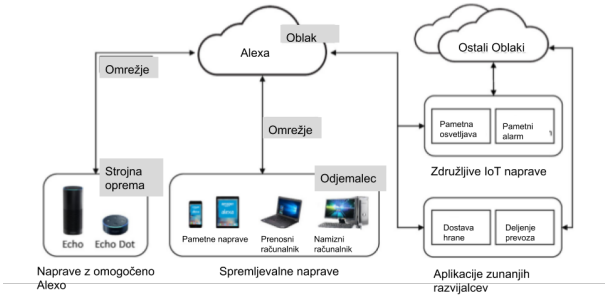
Internet stvari (angl. Internet-of-Things - IoT) se skupaj s tehnologijo vgrajene komunikacije hitro razvija in analitiki v letu 2020 napovedujejo rast trga IoT na 1,7 bilijona [1]. Povsod navzoče pametne naprave bodo ustvarile veliko količino digitalnih podatkov, ki se jih lahko uporabi kot digitalne dokaze. V zadnjem času so preiskovalci in strokovnjaki večkrat poskušali uporabiti "vedno vklopljene" IoT naprave kot vir forenzičnih artefaktov. V članku je naveden primer iz leta 2015, ko je bil James Bates obtožen umora prve stopnje in je policija zasegla pametni zvočnik Echo z omogočeno Alexo [2]. Ko je policija zaprosila Amazon, da jim posreduje koristne podatke glede komunikacije naprave z Alexo, so na Amazonu zavrnili prošnjo, zaradi pomankanja zakonsko obvezujočih zahtev.

Za bolj učinkovito preiskavo podobnih primerov je pomembno razumevanje forenzičnih značilnosti ekosistema Amazon Alexa. Oblačna storitev Alexa se sporazumeva z različnimi napravami z omogočeno Alexo (kot je Amazon Echo), združljvimi IoT napravami in aplikacijami drugih razvijalcev, ter prevaja glasovne ukaze v protokol, ki ga ostale storitve razumejo. Nastavitve okolja Alexa lahko uporabniki upravljajo preko spremljevalnih odjemalcev, kot so mobilne aplikacije in spletni brskalniki. Opisan ekosistem povezanih naprav je v članku poimenovan ekosistem Amazon Alexa.

V članku predlagajo nov forenzični pristop za ekosistem Amazon Alexa, ki združuje forenziko v oblaku in na odjemalcih. Čeprav je pridobitev cloud-native artefaktov zelo pomembna, ima ta dve oviri. Prva ovira je, da je za dostop do podatkov potreben veljaven uporabniški račun in teh ni vedno mogoče pridobiti. Drugo oviro pa predstavljajo izbrisani podatki iz oblaka. V ta namen je pomembno preiskati tudi spremljevalne odjemalce, saj lahko vsebujejo pomembne artefakte.

V članku so predstavili na podlagi analiz razvito orodje za okolja, ki temeljijo na oblačnih storitvah. Orodje imenovano CIFT: Cloud-based IoT Forensic Toolkit lahko pridobi artefakte iz oblaka Alexa s pomočjo neuradnih API klicev in na odjemalcih analizira artefakte povezane s spletnimi aplikacijami. Poskusili so tudi normalizirati podatke v podatkovni bazi in jih vizualno predstaviti. V nadaljevanju je članek razdeljen na poglavja: Amazon Alexa in digitalna forenzika – kjer je predstavljen ciljni sistem; Sorodna dela – kjer so omenjena pretekla dela na tem področju; Artefakti forenzike na ekosistemu Amazon Alexa – kjer so predstavljene ugotovitve; Oblikovanje in implementacija – kjer je predstavljena implementacija; Vizualizacija in vrednotenje – kjer so rezultati ovrednoteni s pomočjo vizualizacije in na koncu Zaključek in nadaljna dela.

Slika 1: Sestavni deli ekosistema Amazon Alexa.



2 Amazon Alexa in digitalna forenzika

V svetu interneta stvari (Internet of Things - IoT) je spodbujano, da si uporabniki sami razvijajo IoT naprave, vendar ker je to zapleteno, si uporabniki naprave kot so: luči, senzorji, pametni pomočniki in podobne, raje kupijo.

Čeprav je na tržišču mnogo podobnih produktov, se v tem članku osredotočamo na Amazon Echo. Pametne naprave iz družine Amazon Echo (Echo, Tap in Dot) so povezane s storitvijo v oblaku Alexa Voice Service (AVS). S pomočjo glasovno upravljane osebne pomočnice Alexa lahko Echo počne mnogo stvari, kot so: predvajanje glasbe, iskanje informacij, upravljanje z naročili in podobno. Med letoma 2015 in 2016 je bilo prodanih več kot 11 milijonov naprav družine Amazon Echo. Poleg tega je bilo razglašeno povezovanje Alexe z različnimi napravami, kot so povezani avtomobili, pametni hladilniki in roboti, kar pomeni, da bi lahko okolje povezano z Amazon Alexo postalo pomemben vir potencialnih digitalnih dokazov.

2.1 Ekosistem Amazon Alexa

Storitev v oblaku Alexa in Echo, ki upravlja z vmesnikom za komunikacijo, predstavljata splošno metodo delovanja IoT produktov, saj jih večina sloni na storitvah v oblaku pri povezovanju s spremljevalnimi odjemalci in združljivimi napravami.

Kot je prikazano na sliki 1, je ekosistem Amazon Alexa sestavljen iz storitve v oblaku Alexa, ostalih oblakov, naprav z omogočeno Alexo, spremljevalnih naprav, združljivih naprav in aplikacij zunanjih razvijalcev. Za komunikacijo s storitvijo v oblaku Alexa so potrebne naprave z omogočeno Alexo, med katerimi se članek osredotoča na naprave Echo. Oblačna platforma Alexa predstavlja vse storitve oziroma operacije, ki jih podpira ekosistem. Med njih spadajo tudi glasovna storitev Alexa, avtentikacija in upravljanje s podatki. Spremljevalne naprave se uporabljajo za dostop do strežnika v oblaku. Storitve v oblaku Alexa se lahko povezujejo tudi z drugimi združljivimi IoT napravami, aplikacijami in storitvami v oblaku.

V nadaljevanju je zaradi omenjenih značilnosti ekosistema Amazon Alexa predstavljen več nivojski forenzični pristop.

2.2 Strojna oprema: naprave z omogočeno Alexo

Za analizo na strojnem nivoju je potrebno napravo razstaviti. V članku je omenjena raziskava, ki analizira Amazon Echo na strojnem nivoju, kjer avtorji obnovijo poskuse za vzratno izdelavstvo (angl. reverse engineering) s pomočjo eMMC Root, JTAG in razhroščevalnih vrat. Vendar avtorji ne omenjajo podrobnosti o podatkih shranjenih na napravi.

2.3 Omrežje: komunikacijski protokol

Naprave z omogočeno Alexo in spremljevalne naprave naj bi komunicirale z Alexo preko interneta, zato so v članku s pomočjo namestnika za spletno razhroščevanje Charles analizirali promet v omrežju. Ugotovili so, da se po ustvarjeni seji z veljavnim identifikatorjem in geslom, večina prometa pomembnega za forenzično raziskavo prenaša preko kriptirane povezav. Z omenjeno analizo so uspeli identificirati cloud-native in odjemalsko-centrične artefakte.

2.4 Oblak: storitve v oblaku Alexa

Alexa je glavna komponenta omenjenega ekosistema in komunicira preko vnaprej določenih API klicev. Ker API klici niso javno objavljeni, so v članku izvedli analizo, da bi odkrili neuradne API klice in pridobili cloud-native artefakte.

2.5 Odjemalec: spremljevalni odjemalci Alexe

Za vzpostavitev naprav z omogočeno Alexo je potreben vsaj en spremljevalni odjemalec, s katerim lahko uporabniki urejajo nastavitve okolja, pregledujejo pretekle pogovore in vklopijo/izklopijo različne možnosti, s pomočjo mobilnih aplikacij ali spletnih brskalnikov. Pri tem se na odjemalce shranijo velike količine podatkov, tako da je izjemno pomembno zajeti odjemalsko-centrične artefakte skupaj s cloud-native artefakti, kar so tudi poskusili storiti v nadaljevanju članka.

3 Sorodna dela

3.1 Internet stvari in digitalna forenzika

V članku omenjajo različne raziskave, med katerimi sta predstavitev scenarijev, kjer je osumljenec uporabljal različne IoT naprave pri kriminalnih dejanjih, in razprava o potencialnih virih digitalnih dokazov. Omenjene so tudi štiri glavne faze pri digitalni forenziki (identifikacija, ohranjanje, analiza in predstavitev), delitev IoT forenzike na forenziko naprav, omrežja in forenziko v oblaku, ter predstavitev ogrodja, ki je sestavljeno iz proaktivnih procesov, IoT forenzike in odzivnih procesov.

3.2 Forenzika v oblaku

Forenzika v oblaku igra ključno vlogo pri uporabi ekosistema Amazon Alexa v digitalni forenziki in obstoječe raziskave jo delijo na dve področji. Prvo področje je forenzika v oblaku, ki temelji na odjemalcih, kjer se pridobi in analizira podatke na odjemalcih, ki jih shranijo aplikacije ali spletni strežniki.

Naslednje področje je cloud-native forenzika, ki pridobiva in analizira podatke iz storitev v oblaku, kot so Dropbox, Google drive in podobni. Omenjen je tudi članek [4] v katerem je avtor razvil orodje za pridobivanje podatkov iz oblaka.

3.3 Predhodne raziskave forenzike Amazon Alexa

V članku so omenjene raziskave, kjer so našli artefakte, shranjene pri uporabi različnih Androidnih mobilnih aplikacij v povezavi z IoT. Pri aplikacijah, ki se tičejo Amazon Echo, so našli SQLite podatkovno bazo in datoteke spletnega predpomnilnika, ki vsebujejo pomembne informacije računov in interakcije z Alexo. V članku je omenjena Bensonova python skripta, ki pridobi podatke v seznamih iz SQLite podatkovne baze za iOS aplikacijo Amazon Alexa, shranjeno na iTunes.

3.4 Vpliv sorodnih dela na smer raziskave

V članku so se na podlagi pregledane literature odločili, da združijo obe področji forenzike v oblaku. Čeprav je cloud-native forenzika ključnega pomena pri analizi vedenja uporabnika, ima ta dve omejitvi, in sicer, da je potrebno veljavno uporabniško ime in geslo, ter je praktično nemogoče obnoviti izbrisane podatke iz oblaka. V ta namen je potrebno poiskati odjemalsko-centrične artefakte, shranjene v spremljevalnih odjemalcih, in z njimi izboljšati cloud-native forenziko. Poleg tega, je pomembno razumeti neobdelane podatke v oblaku.

4 Artefakti forenzike na ekosistemu Amazon Alexa

4.1 Testno okolje

Za potrebe testiranja so vzpostavili testno okolje, ki je vključevalo dva Amazonova produkta Echo Dot in različne odjemalce. Na strani odjemalca sta bili uporabljeni mobilni napravi Android in iOS, ki sta s pomočjo mobilne aplikacije dostopali do storitev, ki jih je ponujala Amazon Alexa. Za testiranje spletne aplikacije je bil uporabljen priljubljeni spletni brskalnik Chrome na operacijskem sistemu Windows in OS X.

4.2 Cloud-native artefakti

Ključ k razumevanju delovanja ekosistema je prepoznavanje podatkov, shranjenih v oblaku. V članku so s pomočjo analiz prometa podatkov identificirali, da je večina prometa prenesenega preko šifriranih povezav, medtem ko so podatki vrnjeni v JSON (JavaScript Object Notation) formatu. Naslednji korak je bil pridobiti podatke s pomočjo veljavnega uporabniškega imena in gesla.

Tako kot tudi ostale storitve v oblaku, ima tudi Amazon Alexa svoj API, ki pa ni javen. Avtorji članka so s pomočjo druge literature pridobili uporabne informacije, s katerimi so lahko odkrili dostopno točko, ki je ponujala konfiguracije in aktivnosti uporabnika.

Rezultati analize podatkov so pokazali, da lahko vsakega izmed API-jev kategoriziramo v sedem različnih skupin in sicer:

- uporabniški račun
- nastavitve uporabnika
- naprava z Alexa sistemom
- združljive naprave
- spretnosti
- aktivnosti uporabnika
- ostalo

Ena izmed zanimivejših ugotovitev, navedenih v članku je dejstvo, da je velik del podatkov vseboval polje s časovnim žigom v formatu UNIX. To nam pove, da je mogoča rekonstrukcija aktivnosti uporabnika s pomočjo časovnega pasu, pridobljenega z API-jem, ki ponuja preference naprave. Prav tako pa so nekateri odgovori strežnika ponujali zadnji del URL-ja, ki je vseboval glasovno datoteko uporabnika v oblaku. To pomeni, da je mogoče željeno datoteko pridobiti na enem izmed API-jev, ki jih ponuja strežnik.

4.3 Odjemalsko-centrični artefakti

4.3.1 Podatkovna baza mobilne aplikacije Alexa

S pomočjo mobilne aplikacije in pametnih mobilnih naprav Android in iOS uporabnik lahko dostopa do Alexinega ekosistema. Na Androidovih napravah aplikacija uporablja dve SQLite datoteki: map_data_storage.db, katera vsebuje informacije o žetonih trenutno prijavljenega uporabnika in DataStore.db, v kateri je moč najti shranjene nakupovalne listke in sezname opravil uporabnika. Pri sistemih iOS aplikacija upravlja z eno datoteko poimenovano LocalStorage.sqlite, ki pa vsebuje zgolj nakupovalne listke in sezname opravil. Pri tem v članku poudarjajo, da so pri pridobivanju datotek prišli do določenih omejitev saj so bile analizirane izključno datoteke, pridobljene iz varnostne kopije iTunesa.

Rezultati preiskave podatkovnih baz so pokazali, da se ne da veliko pridobiti iz podatkov, ki so shranjeni lokalno.

4.3.2 Andoridov WebView predpomnilnik

Amazon Alexa je praktično spletna aplikacija, kar pomeni, da uporablja WebView razred za prikaz spletne vsebine na Androidnih mobilnih napravah. Tako obstaja možnost, da se oblačni artefakti nahajajo v predpomniku shranjenem s strani WebView razreda.

Vsaka predpomnilniška datoteka znotraj Cache imenika je sestavljena iz niza, ki predstavlja URL in podatkovnega toka. Na podlagi prejšnjih raziskav je bilo mogoče ugotoviti, da WebView predpomnilni format vsebuje zaglavje in nogo veliko 8 bajtov kot tudi 4 bajtno polje za shranjevanje velikosti niza.

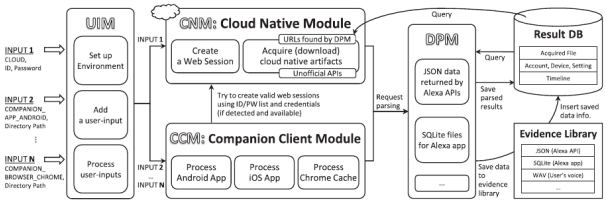
Podatki znotraj podatkovnega toka so bili stisnjeni, kar je pomenilo, da je bilo podatke potrebno razširiti. Komaj takrat so bili podatki pridobljeni v berljivem formatu JSON.

4.3.3 Chrome spletni predpomnilnik

Poleg mobilne aplikacije lahko uporabniki dostopajo do Alexinih storitev na spletni strani. Podatki so tam shranjeni znotraj majhnih bločnih datotek (data_), kar pomeni da je bil potreben pregled prav vseh vnosov predpomnilnika. Vsak vnos je vseboval dva podatkovna tokova, enega za HTTP zaglavje in drugi za shranjene podatke.

Podatki pridobljeni iz Android WebView razreda in Chrome predpomnilnika lahko služijo kot dober digitalni dokaz. Čeprav so podatki iz predpomnilnika lahko zelo koristni, posebno ko nimamo veljavnih podatkov za vpis ali pa ko so bili izbrisani nekateri pomembni artefakti pa ima ta pristop nekaj omejitev. Namreč predpomnilnik se zapolni s podatki zgolj ko uporabnik želi dostopati do izbrane vsebine preko Alexinega API-ja. Prav tako

Slika 2: Pretok.



pa so podatki lahko izbrisani ali prepisani ob katerikoli priložnosti.

4.4 Oblikovanje in implementacija

4.4.1 Oblikovanje CIFT orodja

Tako imenovani CIFT (Cloud-based IoT Forensic Toolkit) je orodje, ki je zasnovan na podlagi rešitve predlagane v povzetem članku. Orodje ponuja vmesnik za lažje forenzično preiskovanje IoT izdelkov. Rešitev je prikazana na sliki 2 in je sestavljena iz 4-ih komponent:

- UIM - user interface module
- CNM - cloud-native module
- CCM - companion client module
- DPM - data-parsing module

UIM ali modul uporabniškega vmesnika nudi vmesnik za nastavljanje okolja, kot tudi procesiranje vnosa uporabnika. Vsak vnos uporabnika je praviloma sestavljen iz operacije in podanih argumentov. V primeru, da se tip operacije nanaša na cloud-native arhitekturo uporabniški vmesnik zahteva isto imenovani cloud-native modul. Nasprotno pa zahteva CCM ali modul za odjemalca.

Za začetek komunikacije z cloud-native modulom je potrebna kreacija spletne seje z izbranim oblačnim sistemom. V primeru povzetega članka je bil to oblačni sistem Alexa. Po uspešni prijavi modul poskuša prenesti podatke s pomočjo neuradnih API-jev. Vrnjeni podatki podani v JSON formatu so nato posredovani modulu za razčlenjevanje podatkov DPM, ki skrbi za shranjevanje podatkov v podatkovno bazo.

4.4.2 Implementacija

V sklopu povzetega članka je bil razvit Python paket, ki vsebuje vse module povzete v prejšnjem poglavju. Kot je vidno na sliki 3 modul uporabniškega vmesnika ponuja tri primarne metode za postavitev okolja in procesiranje vnosov uporabnika.

Slika 3: Primer izvajanja Alexa modulov v CIFT orodju.

Step	Sample codes
Create an instance	`amazon_alexa = AmazonAlexaInterface()`

Set up environments	`amazon_alexa.basic_config(`
	`  path_base_dir="~/CIFT-Result/",`
	`  browser_driver=CIFTBrowserDrive.PHANTOMJS)`

Add inputs	`amazon_alexa.add_input(`
	`  CIFTOperation.CLOUD,`
	`  "**n*h**@*mail.com", "my@1234#password")`
	`amazon_alexa.add_input(`
	`  CIFTOperation.COMPANION_APP_ANDROID,`
	`  "~/Alexa/Android/com.amazon.dee.app/")`
	`amazon_alexa.add_input(`
	`  CIFTOperation.COMPANION_APP_IOS,`
	`  "~/Alexa/iOS/com.amazon.echo/")`
	`amazon_alexa.add_input(`
	`  CIFTOperation.COMPANION_BROWSER_CHROME,`
	`  "~/Alexa/Windows 1/chrome_cache/")`
	`amazon_alexa.add_input(`
	`  CIFTOperation.COMPANION_BROWSER_CHROME,`
	`  "~/Alexa/Windows 2/chrome_cache/")`

Process inputs	`amazon_alexa.start()`

4.5 Vizualizacija in vrednotenje

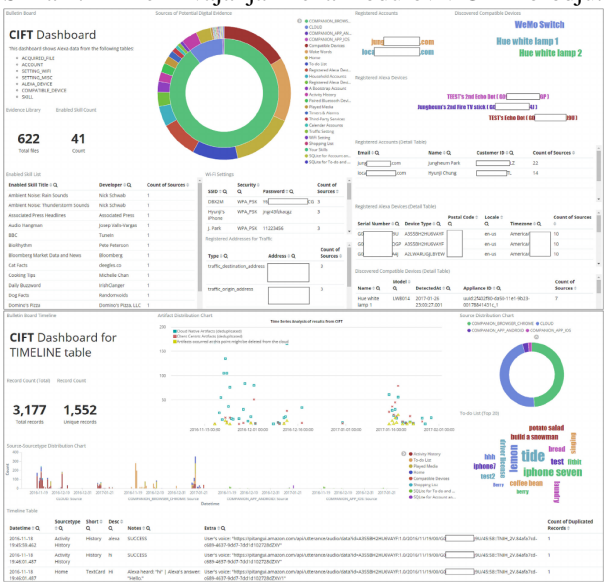
Za iskanje, analizo in vizualizacijo podatkov so avtorji članka uporabili Elastic Stack. To je skupina produktov, ki vključuje 3 znane komponente in sicer Elasticsearch, Logstash in Kibana. Slika 4 prikazuje dve nadzorni plošči za lažjo analizo podatkov. Prva nadzorna plošča vključuje vse tabele razen tabele s časovnico dogodkov. Na levi sta prikazani dve številki, ki prikazujeta število shranjenih datotek in število vključenih spretnosti. Na drugi strani tortni diagram prikazuje razvrščenost različnih virov dokazov v knjižnici dokazov. Nadzorna plošča prav tako vključuje prikaz podrobnosti različnih tabel kot so Wi-Fi nastavitve, uporabniški računi in združljive naprave.

Druga nadzorna plošča je namenjena zgolj tabli s časovnico. Tako je mogoče videti kronološki potek dogodkov, ki se lahko navezujejo na cloud-native (označeni s kvadratom) ali odjemalsko-centrične artefakte (označeni s križem). Dogodki označeni z trikotnikom označujejo artefakte, ki so bili morda izbrisani. Prav tako pa vsebuje tudi tabele z podatkovnimi zapisi, kot tudi prikaz seznama opravil in nakupovalnih listkov.

5 Zaključek in nadaljnje delo

V vsakdanjem svetu se IoT naprave uporabljajo vse pogosteje. Z njihovo uporabo se kopiči število podatkov tako na oblačnih kot na lokalnih sistemih. Zato je toliko bolj pomembna analiza podatkov v sistemih na katere je povezana preiskana naprava. Forenzična rešitev predlagana v članku je ena izmed prvih, ki vključuje Amazonov produkt Echo in njegov ekosistem. Rešitev združuje cloud-native in odjemalsko-centrične forenzične artefakte. Poleg rešitve je bilo tudi implementirano orodje CIFT, ki uporabniku omogoča pridobitev artefaktov iz

Slika 4: Primer izvajanja Alexa modulov v CIFT orodju.



sistema Alexa in analizo lokalnih artefaktov prejetih od odjemalcev sistema.

Avtorji povzetega članka imajo željo razširiti možnost uporabe na ostale IoT produkte in se tako znebiti omejitve delovanja sistema zgolj na produktih z Alexa sistemom. Njihov cilj je prav tako razvoj novih komponent orodja CIFT, ki bi omogočale boljše in lažje delo.

References

[1] Explosive internet of things spending to reach $1.7 trillion in 2020, according to idc. https://www.businesswire.com/news/home/20150602005329/en/, 2015. [Online; accessed 10-May-2018].

[2] An Amazon Echo may be the key to solving a murder case. https://techcrunch.com/2016/12/27/an-amazon-echo-may-be-the-key-to-solving-a-murder-case/, 2016. [Online; accessed 10-May-2018].

[3] CHUNG, H., PARK, J., AND LEE, S. Digital forensic approaches for amazon alexa ecosystem. *Digital Investigation 22* (2017), S15–S25.

[4] V. ROUSSEV, A. BARRETO, I. A. Api-based forensic acquisition of cloud drives. *Adv. Digital Forensics XII* (2016), 213–235.

DROP (DRone Open source Parser): Forenzična analiza modela DJI Phantom III

Rok Plevel
Univerza v Ljubljani
Fakulteta za računalništvo in informatiko
rok.plevel@gmail.com

Matej Pečnik
Univerza v Ljubljani
Fakulteta za računalništvo in informatiko
matej@pecnik.si

Gal Kos
Univerza v Ljubljani
Fakulteta za računalništvo in informatiko
gal.kos@outlook.com

ABSTRACT

V sklopu tega dela smo analizirali članek *DROP (DRone Open source Parser) your drone: Forensic analysis of the DJI Phantom III* avtorjev Clark et al.[3].

Brezpilotno letalo *DJI Phantom III* je bilo v letih 2016 in 2017 že uporabljeno pri zlonamernih dejavnostih. V času pisanja analiziranega članka je bilo podjetje DJI proizvajalec z največjim tržnim deležem na področju brezpilotnih letal. Avtorji Clark et al. so predstavili forenzično analizo modela *DJI Phantom III* ter implementirali razčlenjevalnik za strukture datotek, ki jih hrani preiskovano brezpilotno letalo.

V obravnavanem članku so avtorji predstavili odprtokodno orodje *DRone Open source Parser (DROP)*, ki je namenjeno lastniškim datotekam DAT pridobljenih iz notranjega pomnilnika brezpilotnega letala. Analizirani članek vsebuje predhodne ugotovitve o datotekah TXT, ki se nahajajo na mobilni napravi, ki upravlja in nadzoruje brezpilotno letalo. Z izločanjem podatkov iz nadzorovalne mobilne naprave in brezpilotnega letala so avtorji Clark et al. korelirali podatke ter na podlagi pridobljenih podatkov povezali uporabnika z določeno napravo. Poleg tega so rezultati analiziranega članka pokazali, da je najboljši mehanizem za forenzično pridobivanje podatkov iz brezpilotnega letala, ročna odstranitev kartice SD.

Ugotovitve avtorjev so pokazale, da brezpilotno letalo ne sme biti ponovno vklopljeno, saj ponovni prižig letala spremeni podatke z ustvarjanjem nove datoteke DAT in lahko izbriše shranjene podatke, če je notranji pomnilnik brezpilotnega letala poln.

Keywords

računalniška forenzika, digitalna forenzika, forenzika IoT, forenzika vgrajenih sistemov, forenzika UAV, forenzika brezpilotnih letal, forenzika mobilnih naprav, brezpilotno letalo, dron, UAV, DJI, Phantom III, DJI Phantom III, Open source DRone Parser, struktura datoteke DAT, struktura datoteke TXT

1. UVOD

Brezpilotna letala (angl. *Unmanned Aerial Vehicle* - UAV), imenovana tudi droni, so v zadnjih letih postala vse bolj priljubljena. Glavne razloge za povečanje priljubljenosti lahko iščemo predvsem v hitrem povečanju dostopnosti za povprečnega uporabnika. Letala UAV niso več drage naprave in jih je pogosto mogoče enostavno upravljati, na primer z uporabo aplikacije na pametnem telefonu, ki omogoča ali neposreden nadzor ali prikazovanje povratnih informacij.

Kitajsko podjetje Da-Jiang Innovations Science and Technology (DJI) je danes eden največjih proizvajalcev letal UAV. Članek na spletni platformi *dronelife*[1] poroča, da so se letni prihodki podjetja DJI med letoma 2011 in 2013 povečali iz 4,2 milijona na 130 milijonov dolarjev. Od začetka proizvodnje do konca leta 2014 je bilo prodanih več kot 500.000 brezpilotnih UAV. Podjetje DJI proizvaja dva primarna modela letala, in sicer serijo *Phantom* in serijo *Inspire*. Predmet obravnavane raziskave je bila forenzična analiza modela *Phantom III Standard*, ki je bil lansiran aprila 2015.

S povečanjem priljubljenosti letal UAV se je povečala tudi potreba po zakonih o uporabi letal. Zvezna uprava za letalstvo Ministrstva za promet Združenih držav Amerike (*Federal Aviation Administration* - FAA) je od avgusta 2015 do januarja 2016 poročala o 583 nesrečah v katerih so bila vpletena letala UAV. Ti incidenti so po navadi vključevali nedovoljene polete v omejen zračni prostor. Ameriški Zvezni preiskovalni urad (*Federal Bureau of Investigation* - FBI) je bil zadolžen za preiskavo neprijatelskih letal UAV, pri čemer je za najzahtevnejšo naloga veljalo predvsem sledenje krivde pilota[2]. Da bi sledili tehnologiji, si administracija FAA prizadeva razviti zakone, ki bi omejili rabo rekreativnih letal UAV. Ti zakoni večinoma omejujejo najvišjo nadmorsko višino in uvajajo območja omejenega zračnega prostora za letala UAV, kot so na primer letališča ali večji dogodki.

Letala UAV so vplivala tudi na terorizem in z njim povezane dejavnosti. Islamska država Iraka in Levanta (ISIL) na primer že nekaj časa uporablja letala UAV za videonadzor[4]. Nadalje so bili zabeleženi trije incidenti države ISIL, ko so bila letala UAV opremljena z eksplozivi. Teroristične skupine pri svojih aktivnostih ne uporabljajo vojaških letal UAV, temveč se zatečejo k komercialno razpoložljivim brezpilotnim letalom, ki vključujejo tudi serijo *DJI Phan-*

tom. Zaradi naraščajoče priljubljenosti komercialnih letal UAV in vzpona kriminalnih dejavnosti, ki vključujejo uporabo le-teh, je potrebno razviti zadovoljive forenzične tehnike za letala UAV. Predpostavimo lahko, da se bo potreba po forenziki letal UAV še naprej povečevala, ko bodo letala UAV postala dostopnejša in uporabljena tudi v drugih kriminalnih dejavnostih.

V okviru te raziskave smo analizirali članek *DROP (DRone Open source Parser) your drone: Forensic analysis of the DJI Phantom III* avtorjev Clark et al.[3], ki predstvlja prvo celovito delo, ki zajema forenzično analizo brezpilotnega letala *DJI Phantom III Standard*. Njihovo delo je zajemalo naslednje sklope:

- predstavili so niz postopkov, ki jih lahko preiskovalci uporabijo med postopkom preiskave primera, ki vključuje brezpilotno letalo *DJI Phantom III Standard*,
- predstavili so javno dostopen račun za strukturo binarne datoteke, ki jo ustvari letalo UAV in shrani na svojo kartico SD,
- njihove ugotovitve so združili v odprtokodno orodje imenovano *DRone Open source Parser* (DROP), ki omogoča forenzično ustrezno razčlenjevanje zgoraj omenjene binarne datoteke,
- zagotovili so račun, orodje in metodo za koreliranje podatkov, pridobljenih iz notranjega pomnilnika letala UAV in mobilne naprave, ki omogoča njegovo nadzorovanje.

1.1 Obseg raziskave

Potrebno je omeniti, da je bil obseg raziskave avtorjev članka omejen na brezpilotno letalo *DJI Phantom III Standard*. Idealno bi bilo preizkušanje več različnih modelov UAV, vendar je bilo delo po poročanju avtorjev precej dolgočasno in je zahtevalo veliko obratnega inženirstva. Prav tako je po mnenju avtorjev članka potrebno opozoriti, da bi bila implementacija splošne rešitve, ki bi omogočala forenzično analizo vseh letal UAV, ki so na voljo na potrošniškem trgu, precej zahtevna ali celo nemogoča naloga, saj je vsako letalo UAV različno glede na operacijski sistem, shranjevanje podatkov in protokole za upravljanje. Avtorji so izbiro modela za preiskavo upravičili z visokim deležem podjetja DJI na trgu brezpilotnih letal in dejstvo, da so njihova letala že uporabljale teroristične skupine, kot je ISIL.

2. METODOLOGIJA

Avtorji so raziskavo izpeljali v naslednjih korakih:

Ponastavitev tovarniške opreme: Da so avtorji zagotovili, da nobena zunanja spremenljivka ni vplivala na rezultate, je bila najprej izvedena tovarniška ponastavitev in formatiranje vseh naprav in kartic.

Oblikovanje scenarija: Po vklopu letala UAV sta bila opravljena dva poleta do ločenih geografskih lokacij, nato so avtorji letalo izklopili.

Zbiranje podatkov: Postopek zbiranja podatkov je bil razdeljen na tri dele, in sicer na zbiranje podatkov iz brezpilotnega letala, iz krmilnika in iz pametnega telefona oziroma tablice.

Analiza podatkov: Pridobljeni podatki so bili analizirani. Pridobljeni sta bili dve še posebej zanimivi datoteki, ki sta vsebovali podatke o letu. Nato je bila izvedena poglobljena analiza zgornjih datotek in njunih struktur.

Implementacija orodja: Ko so bili podatki analizirani in definirane strukture datotek, je bilo implementirano orodje, ki omogoča preiskovalcem razčlenitev dokaznih datotek.

Testiranje: Na koncu so avtorji članka izvedli številne teste, s katerimi so potrdili konstruirano orodji in izsledke raziskave.

2.1 Tovarniška ponastavitev

Prvi korak v procesu je bila tovarniška ponastavitev letala UAV in tabličnega računalnika *Nexus 7*. Tablični računalnik je bil tovarniško ponastavljen s pomočjo menija za obnovitev naprave. Nato je bil tablični računalnik posodobljen na (v času raziskave) najnovejši operacijski sistem Android (6.0.1).

Zatem je bilo tovarniško ponastavljeno še brezpilotno letalo s pomočjo druge naprave Android z nameščeno aplikacijo DJI GO. Aplikacija DJI GO je aplikacija za operacijski sistem Android, ki jo je razvilo podjetje DJI in služi za upravljanje in nadzor brezpilotnega letala *Phantom III*. Aplikacija uporabniku prav tako omogoča, da iz notranjega pomnilnika brezpilotnega letala izbriše podatke o letu in predpomnilnik videa.

Dodatno je bila formatirana zunanja kartica SD, ki je bila nameščena na snemalni napravi na brezpilotnem letalu.

2.2 Oblikovanje scenarija

V drugem koraku je bila najprej nameščena aplikacija DJI GO na primarni tablični računalnik in izvedena sta bila dva testna leta. Oba leta sta bila dokumentirana; avtorji so spremljali datum in čas ter vzorce leta. Podatki so bili zabeleženi z uporabo brezpilotnega letala in ročno s strani raziskovalcev, da so bili upoštevani vsi dogodki v letu.

2.3 Zbiranje podatkov

2.3.1 Pretvarjanje zunanje kartice SD v slikovno datoteko

Ko sta bila zaključena preizkusna leta, so avtorji pričeli s pridobivanjem podatkov. Začeli so s fizično sliko kartice SD, ki jo uporablja kamera *Gimbal* za shranjevanje slik in videoposnetkov. Kartico SD so avtorji odstranili iz kamere ugasnjenega brezpilotnega letala in jo vstavili v blokator pisanja *Cellebrite*. Nato so izračunali in shranili zgoščevalno vrednost funkcije MD5 in s pomočjo ukaza *disk dump* (dd) pretvorili celotni disk v slikovno datoteko. Nazadnje so vrednost zgoščevalne funkcije MD5 izračunali še nad slikovno datoteko in obe vrednosti primerjali ter datoteki verificirali. Kasneje je bila slikovna datoteka odprta v orodju *FTKImager 3. 1. 1*, kjer je bila vsebina pregledana in izvlečena v drug direktorij. Izvlečene datoteke so vključevale slike in videoposnetke, ki so bili posneti med leti ter nekatere datoteke z metapodatki o videoposnetkih.

2.3.2 Varnostna kopija sistema Android

Nato so avtorji pričeli z delom na tabličnem računalniku *Nexus 7*, ki je deloval kot kontrolna postaja in nadzorna plošča za brezpilotno letalo. Aplikacija DJI GO, ki je bila nameščena na tablični računalnik je omogočala prikaz žive kamere, podatkov o bateriji, GPS in omogočala uporabniku izdajanje ukazov, kot sta avtomatsko vzletanje in pristajanje. Varnostne kopije v operacijskem sistemu Android vsebujejo predvsem aplikacijske artefakte in tipično zagotavljajo podatke o lastništvu in uporabi aplikacij (tudi aplikacije DJI GO). Logična varnostna kopija je bila izvedena z uporabo orodja *adb backup-all*. Orodje *Android Backup Extractor* je bilo uporabljeno za pretvorbo varnostne kopije v datoteko *.tar*. Nazadnje so avtorji s pomočjo orodja *7zip* datoteko *.tar* dekompresirali v nov imenik. Datoteke vsebovane v tem imeniku so se nanašale na vsako aplikacijo, ki je bila nameščena na tabličnem računalniku vključno z datotekami aplikacije DJI GO. Avtorji so prav tako izpostavili, da bi bilo kreiranje fizične slike naprave Android nepotrebno, saj so želeli iz naprave pridobiti le logične podatke.

2.3.3 Pomnilnik sistema Android

Medtem ko je bila varnostna kopija sistema Android uporabna pri pridobivanju podatkov o aplikacijah, je bilo potrebno uporabniške podatke pridobiti iz notranjega pomnilnika mobilne naprave. To so avtorji naredili tako, da so napravo povezali s forenzično delovno postajo in vse datoteke shranjene v notranjem pomnilniku prekopirali na delovno postajo. Notranji pomnilnik je vseboval več direktorijev, ki so se nanašali na aplikacijo DJI in so bili kasneje analizirani.

2.3.4 Pridobivanje podatkov brezpilotnega letala

Zadnja stopnja pridobivanja podatkov je bilo pridobivanje zapisov letov iz notranjega pomnilnika brezpilotnega letala. To so avtorji izvedli z uporabo treh različnih metod:

1. S pripenjanjem notranjega pomnilnika brezpilotnega letala na forenzično delovno postajo preko aplikacije DJI GO in nato z ročnim kopiranjem datotek na forenzično delovno postajo. Avtorji so opazili, da je bil notranji pomnilnik pripet na način, ki je omogočal samo pravice za branje. Testiranje je pokazalo, da ta metoda morda ne bi bila forenzično ustrezna, saj je moralo biti brezpilotno letalo med postopkom vklopljeno.
2. Druga metoda je bila enaka prvi z razliko, da tu pridobimo fizično sliko notranjega pomnilnika s pomočjo ukaza *dd*. Pri tem je potrebno upoštevati, da veljajo enake omejitve kot pri prvi metodi.
3. V sklopu zadnje metode je bil dejanski medij notranjega pomnilnika izvlečen iz brezpilotnega letala. To je vključevalo razstavitev brezpilotnega letala, odklop več žic in odstranitev lepila, ki je bilo namenjeno trajnemu fiksiranju notranje kartice SD. Preizkusi avtorjev so pokazali, da je to forenzično najustreznejša metoda za pridobivanje podatkov notranjega pomnilnika brezpilotnega letala.

3. ANALIZA PODATKOV

Na dronu DJI Phantom III najdemo dva primarna vira, ki hranita podatke o letenju. Datoteko TXT ustvarjeno z mobilno aplikacijo (datoteka je shranjena na mobilni napravi) in datoteko DAT ustvarjeno na dronu. DAT datoteko najdemo na dronu na notranjem pomnilniku.

Obe datoteki sta šifrirani in kodirani z uporabo dveh različnih formatov. Z dekodiranjem teh dveh datotek lahko pridobimo podatke o GPS, motorjih, daljinskem upravljalniku, statusu letenja in še nekaj ostalih informacij. Te datoteke služijo kot elektronski zapisovalnik o poletih drona.

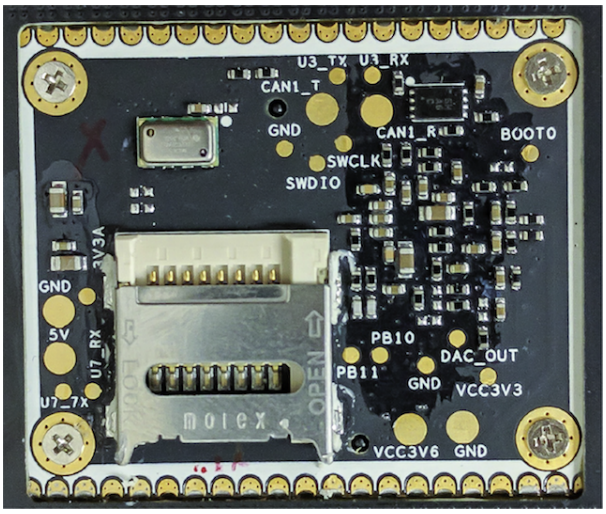


Figure 1: Izvlečena notranja SD kartica brezpilotnega letala Phantom III s spodnjega dela matične plošče.

3.1 Aplikacija DJI GO - zapisi o letenju drona iz datoteke TXT

Na Android napravi na kateri smo upravljali brezpilotno letalo preko aplikacije DJI GO lahko najdemo več datotek TXT, ki vsebujejo podatke o letih. Te datoteke lahko najdemo v *InternalStorage/DJI/ dji.pilot/FlightRecord/*. Datoteke so poimenovane kot *YYYY-MM-DD_[HH-MM-SS].txt* pri čemer datum in čas datoteke predstavlja datum in čas začetka leta. Te datoteke vsebujejo podatke o lokaciji, statusu letenja, bateriji in drugo. Podatki so shranjeni v obliki paketov. Ti paketi morajo biti dekodirani in dešifrirani.

3.2 Aplikacija DJI GO - struktura datoteke TXT

Avtorji članka so s pomočjo obratnega inženirstva aplikacije DJI GO, ki ji bila nameščena na mobilni napravi z operacijskim sistemom Android, poskusili ugotoviti strukturo datoteke TXT. Aplikacijo DJI GO so prenesli iz uradne spletne strani ter jo povratno prevedli z uporabo grafičnega programa za povratno prevajanje datotek Java, JD-GUI. S tem so pridobili to, da so vsi razredi aplikacije postali vidni. Čeprav je veliko spremenljivk, funkcij in imen razredov bilo zapletenih, so iskali po ključnih besedah. Iskanje po ključni besedi "FlightRecord" (ime direktorija, kjer so shranjene datoteke TXT) jih je naslovilo na razred *k.class*, ki se nahaja znotraj paketa *dji.pilot.fpv.model* , ki je del *classes2.jar*. Razred *k.class* je zadolžen za pisanje zapisov o letu. Z ročnim

pregledom postopka pisanja datotek TXT v Android aplikacijo so s pomočjo urejevalnika hex editor ugotovili, da datoteka sledi neki strukturi. Strukturo datoteke prikazuje slika 2. Pomembno je vedeti, da so podatki v teji datoteki zapisani po pravilu tankega konca (angl. little endian rule). Zadnjih 190 bajtov datoteke vsebuje splošne informacije o zračnem plovilu in letu. Prvi del informacij je opcijska informacija z imenom geotag, ki je shranjena kot golo besedilo npr. “New Haven, Connecticut.”. Naslednji bit prepoznanih podatkov pride nekaj bajtov kasneje in predstavlja ime v golem besedilu kot je prikazano tukaj:

Yuhe’s Phantom3
03Z0600080
CL03021337
05LD102XHR
1153516293

To je ime skupaj z imenom modela, ki ga je uporabnik navedel med nastavljanjem nastavitvenega postopka, pri čemer ”Yuhe”predstavlja ime lastnika tega brezpilotnega letala. Imenu sledijo štirje nizi, ki predstavljajo identifikatorje. Iste štiri identifikatorje lahko najdemo tudi v tabeli *dji_pilot_publics_model_DJIDeviceInfoStatModel*, ki se nahaja v podatkovni zbiriki *dji.db*. Avtorji so s pomočjo programske opreme JD-GUI ugotovili, da ti identifikatorji predstavljajo serijske številke, ki jih lahko povežemo s strojnimi napravami.

Prva serijska številka pripada inercialno merilni enoti (angl. Inertial Measurement Unit - IMU), ki jo najdemo znotraj brezpilotnega letala. Druga serijska številka ČL03021337” predstavlja kamero. Tretja serijska številka pa pripada primarnemu vezju, ki se nahaja znotraj daljinskega upravljalnika katerega glavna naloga je nadzor drona. Ta del je odgovoren za prenos podatkov na zaslon mobilne naprave. Zadnja serijska številka ”1153516293”pa pripada bateriji.

3.3 Struktura paketa TXT

Zapisi o letenju drona se v obliki paketov hranijo v datoteki TXT, katerega strukturo prikazuje slika 3. Vsebina paketa je šifrirana in kriptirana. V začetni fazi nastanka članka (citat) še ni bilo objavljenega postopka dekriptiranja te vsebine.

V procesu povratnega prevajanja Android aplikacije DJI GO pa so avtorji našli knjižnico *DJI GO_v2.9.1_apkpure.com 2/lib/armeabi-v7a/libFREncry pt.so*, ki se uporablja za šifriranje in dešifriranje podatkov o letenju, ki so shranjeni v datoteki TXT. Avtorjem članka je s pomočjo orodja IDA-Pro uspelo izolirati funkciji za šifriranje in dešifriranje. Razumevanje vhodnih parametrov funckij pa je trenutno še v delu.

3.4 DJI - datoteke DAT

Datoteke DAT najdemo v notranjem pomnilniku drona. Vse datoteke DAT so poimenovane kot *FLY###.DAT* pri čemer “###” predstavlja zaporedno številko. Ta vrsta datoteke vsebuje veliko podatkov povezanih z letom (lokacija drona, status letenja in odčitki iz različnih senzorjev).

3.5 DJI - struktura datoteke DAT

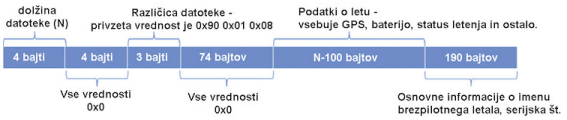


Figure 2: Struktura datoteke TXT.

Figure 3: Struktura paketa TXT.

Po ekstrahiranju datotek iz SD katice brezpilotnega letala (datotečni sistem FAT32), so avtorji članka poskusili prebrati datoteko DAT. Hitro je bilo jasno, da je datoteka kodirana in posledično zahteva dekodiranje. Pri raziskovanju je bila odkrita kratka uradna dokumentacija v povezavi z DJI-jevimi datotekami DAT. A vseeno obstaja veliko pogovorov ter orodij iz strani ljubiteljev, ki poskušajo dekodirati datoteke DAT. Najbolj zadovoljiv izhod iz datoteke daje orodje z imenom *DatCon*, vendar vsa polja še vseeno niso dekodirana. Orodje DatCon omogoča razčlenjevanje podatkov iz binarne datoteke DAT in izvoz teh podatkov v človeku bolj prijazno berljivo datoteko CSV. DatCon program je bil prenesen kot izvršljiva datoteka jar. Z povratnim prevajanjem datoteke jar je bilo mogoče pridobiti bolj celovito razumevanje strukture datotek DAT. Strukturo datoteke DAT prikazuje slika 4. Struktura datoteke je preprosta. Prvih 128 bajtov datoteke predstavlja glavo podatkov. Bajti od 16-20 vsebujejo besedo ”BUILD”, sledi datum in čas izdelave datoteke. Točen pomen polja datum izdelave ni polnoma znan, povsem verjetno naj bi se nanašal na datum zadnje spremembe nad datoteko. Orodje za razčlanjevanje podatkov, ki ga bomo pojasnili v naslednjem poglavju uporablja besedo ”BUILD”, kot indikator, saj naj bi se na ta način resnično prepričali, da je datoteka, ki jo beremo DJI datoteka DAT. Bajt 128 predstavlja začetek podatkovnih paketov. Podatkovni paketi so v datoteki zapisani v obliki binarne strukture, ki so v lastništvu DJI. Večino datoteke vsebuje pakete te vrste. Konec datoteke pa označuje niz bajtov, ki vsebujejo ničle.

3.6 DJI - struktura paketa

Podatkovni paketi so strukture različnih dolžin odvisno od tipa podatkov, ki so vpisani. Čeprav so različne dolžine sledijo skupni strukturi Slika 5 prikazuje osnovno strukturo podatkovnega paketa. Dolžina paketa predtavlja dolžino celotnega paketa vključno z začetnim in kočnim bajtom v uporabni vsebini. Štirje bajti, ki jih najdemo po sporočilnem bajtu predstavljajo notranjo uro vodila. Glavni podatki pa so lahko kjerkoli in so lahko veliki od 14 do 245 bajtov. Prav na ta način lahko med sabo razlikujemo pakete ter na ta način razberemo tip in podtip paketa.

3.7 Stuktura paketa glavnih podatkov

Paket glavnih podatkov je del paketa, ki vsebuje podatke o aktualnih senzorjih ter podatke o telemetriji. Glavni podatke lahko najdemo kjerkoli, dolgi pa so lahko od 14-245 bajtov, ločimo pa jih glede na tip podatkov. Kot prikazuje slika poznamo devet različnih tipov paketa (GPS, motor, domača točka, daljinski upravljalnik, lokacija mobilne naprave, baterija, položaj, status letenja in napredni podatki o bateriji). Podatki o GPS-u vsebujejo podatke o lokaciji brezpilotnega letala, nadmorski višini, 3-osni pospešek, žiroskop, hitrost, magnetometer in druge. Na koncu najdemo štiri bajte podatkov za katere avtorji niso znali določiti namena. Na osnovi kode *DatCon*, podatki o motorju vsebujejo tako podatke o hitrosti kot tudi podatke o obremenitvi vseh štirih motorjev na letalu. Model brezpilotnega letala Phantom III Standard teh podatkov ne vsebuje v svojih paketih. Lokacija mobilne naprave je posredovana samo v načinu Sledi me". To je način avtomatskega pilota, ki ga lahko omogočimo v aplikaciji DJI GO. S tem dosežemo, da nas bo letalo sledilo. Domačo točko ponavadi nastavimo avtomatsko preko aplikacije DJI GO vendar lahko to točko uporabnik nastavi tudi ročno. Koordinate o domači točki najdemo v glavnih podatkih z imenom "home point". Stanje daljinskega upravljalnika, kot so plin, krmilo in dvigalo se beležijo v glavnih podatkih z imenom "remote control". Poznamo dva tipa paketov, ki služita informaciji o bateriji (baterija in napredna baterija). Prvi vsebuje informacije o ravni baterije medtem ko drugi vsebuje bolj podrobne informacije o bateriji kot so kapaciteta, temperatura, tok in napetost posamezne celice.

Zadnja dva tipa glavnih podatkov pa vsebujeta podatke o poziciji kameri in stanju celotnega leta. Stanje leta opisuje podatke o stanju leta (avtopilot, vzlet, pot domov, itd.) ter napakah GPS. Ta paket vsebuje tudi čas leta v milisekundah (od začetka leta).

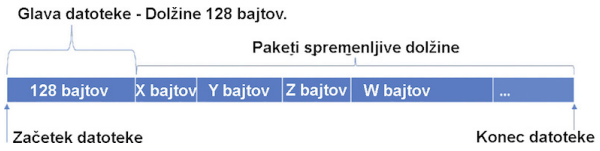


Figure 4: Osnovna struktura datoteke DAT.

4. IZDELAVA ORODJA: DRONE OPEN SOURCE PARSER (DROP)

DROP, oziroma *DRone Open source Parser* je orodje izdelano z Python 3.4, ki obdela DAT datoteko iz spominske kartice brezpilotnega letala DJI Phantom III. Orodje je dobro testirano za obravnavan model, torej DJI Phantom III Standard, vendar deluje tudi pri večini ostalih Phantom modelov tega proizvajalca.

Orodje ima dve osnovni funkciji:

- Račlenitev podatkov iz DJI DAT datotek
- Korelacija DAT datotek z TXT datotekam na Android krmilni napravi

Obdelani podatki se zapišejo v datoteko CSV formata. Koda

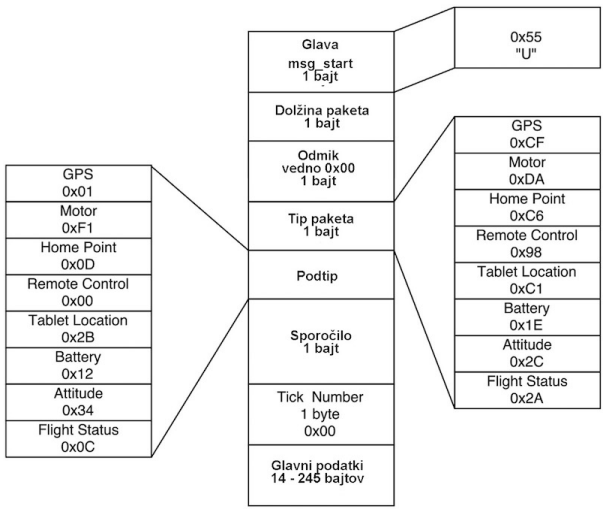


Figure 5: Struktura datoteke DAT.

programa je na voljo javnosti preko Github repozitorija na `https://github.com/unhcfreg/DROP`.

4.1 Razčlenjevanje DAT datoteke

Orodje začne s pregledovanjem metapodatkov DAT datoteke, predvsem, da pridobi velikosti datoteke. Sledi z izločanjem prvih 128 bajtov glave in med bajti 16 in 20 poišče niz *BUILD*. Preko vrednosti niza in metapodatkov določi, da gre za ustrezno DJI DAT datoteko ter nadaljuje z zajemom paketov po 128 bajtov do konca datoteke. Z instanco *Message* prebere glavo in telo datoteke, ki predstavlja različne podatke o stanju brezpilotnega letala ob zapisu paketa. Značilnost vsakega paketa je časovni zapis dogodka (t.i. *tick number*).

Mnogo paketov ima enak časovni zapis, torej so bili paketi zajeti ob istem času, vendar imajo različno vsebino. Paketi so posodobljeni v različnih časovnih intervalih kar pomeni, da za vsak časovni zapis ne pridobi vseh možnih tipov podatkov, ki se beležijo.

Telo paketa je potrebno tudi dešifrirati, kot bo podrobneje opisano v naslednjem poglavju. Paketov ni nujno vedno možno razčleniti, saj so lahko delno ali popolnoma uničeni oziroma ne sledijo formatu DJI DAT datoteke. Stanja paketov in nekatere statistične podatke program *DROP* shrani v zapisniku *processlog.txt*. Psevdo koda glavne funkcionalnosti orodja *DROP* je prikazana na Algoritmu 1.

4.2 Dešifriranje in razčlenitev paketov

Dešifriranje in razlčenitev je z obratnim inžiniringom povzeta po DatCon algoritmu in prikazana v Algoritmu 2. Generira ključ za dešifriranje, ki je zgolj izračun časovnega zapisa (*tick number*) po modulu 256. Vsak bajt je XOR-an s ključem in dodan dešifriranemu sporočilu preko instance *Message*. Opaziti je, da je dešifrirni algoritem pri Phantom DJI modelih zelo preprost in ni jasno zakaj so se proizvajalci odločili za takšno shemo. Po končanem dešifriranju so podatki predstavljeni po ustrezni strukturi. Poleg osnovnih

Algorithm 1 DAT packet parsing algorithm

```
1:  procedure DROP(inputFile, outputFile)
2:      meta ← inputFile metadata
3:      in_file ← open inputFile in read binary mode
4:      file_header ← read bytes 0 to 127 of in_file
5:      build ← unpack file_header[16:21] to string
6:      if build != "BUILD" then
7:          exit()
8:      out_file ← open outputFile in write mode
9:
10:     byte ← read next byte from in_file
11:     message ← new Message()
12:     while byte length != 0 do
13:         if byte != 0x55 then
14:             byte ← read next byte from in_file
15:         else
16:             packetLength ← read next byte from in_file
17:             padding ← read next byte from in_file
18:             if padding == 0x00 & packetLength >0 then
19:                 header ← read next 7 bytes from in_file
20:                 payload ← read packetLength - 10 bytes from in_file
21:                 thisPacketTickNo ← unpack header[3:7] to integer
22:                 if message.tickNo == NULL then
23:                     message.tickNo ← thisPacketTickNo
24:                 if thisPacketTickNo != message.tickNo then
25:                     if message.addedData == TRUE then
26:                         out_file ← message.getRow()
27:                         message.addedData ← FALSE
28:                 message.addPacket(packetLength, header, payload)
29:                 byte ← read next byte from in_file
30:             else
31:                 byte ← padding
32:     out_file ← message.getRow()
33:     in_file.close()
34:     out_file.close()
35:     return
```

Figure 6: Algoritem 1: Obdelava DJI DAT datotek

Algorithm 2 DAT payload decrypt algorithm

```
1:  procedure DECRYPT(payload, tickNo)
2:      xorKey ← tickNo MOD 256
3:      decryptPld ← []
4:      for byte in payload do
5:          append (byte XOR xorKey) to decryptPld
        return decryptPld
```

Figure 7: Algoritem 2: Dešifriranje in razčlenitev

Algorithm 3 DAT to TXT file correlation

```
1:  procedure CORRELATION(dat_data, csv_data)
2:      ft_matches ← 0
3:      gps_matches ← 0
4:      for ft in dat_data do
5:          if ft in csv_data then
6:              ft_matches ← ft_matches + 1
7:              if dat_data[ft][lat] matches csv_data[ft][lat] then
8:                  if dat_data[ft][lon] matches csv_data[ft][lon] then
9:                      gps_matches ← gps_matches + 1
10:     if ft_matches > 0 then return (gps_matches/ft_matches) * 100
        return 0
```

Figure 8: Algoritem 3: Korelacija DJI DAT in TXT datoteke iz mobilne naprave

podatkov kot na primer GPS lokacija ob zajemu, vsebuje tudi informacije, ki niso direktno povezane z prvotnim namenom paketa. Nekaj primerov takšnih podatkov so napetost in tok, vrednosti pospeška na vseh treh oseh in podobno.

4.3 Korelacija s TXT datoteko

Dodatna funkcionalnost orodja je korelacija z TXT datotekmi najdenimi na Android mobilni napravi, ki letalo krmili. Prikaza je na algoritmu 3. Preverja ujemanje podatkov o GPS lokaciji (longitude, latitude, na 5 decimalk natančno), pri zapisih, ki imajo enak časovni zapis. Izkazalo se je da je ujemanje skoraj popolno.

4.4 Integriteta datotek in robustnost orodja

Pravila digitalnih forenzičnih postopkov zahtevajo, da se DAT datoteke niso nikakor spremenile. V ta namen je nad datotekami izračunanih več zgoščevalne vrednosti (MD5, SHA-1, SHA-256) pred in po obdelavi z orodjem *DROP* in tudi *DatCon*.

Forenzična orodja morajo biti dokazano zanesljiva. Robustnost so preizkušali s seti pokvarjenih DAT datotek, pridobljenih s načrtno predelavo dejanskih zajetih datotek. Pokvarjene datoteke so vsebovale na primer odstranjene bajte iz glave ali vsebine, spremenjene začetne bajte in podobno. DatCon se je izkazal za precej manj zanesljivo kar se tiče zajema podatkov in pridobitve ustrezne vsebine. Preverjena je tudi pravilnost korelacije med DatCon TXT datotekami in DAT datoteki na napravi.

Tudi nad dejansko SD kartico je bilo izvedenih nekaj preizkusov. Raziskali so predvsem učinke ob dosegu največje kapacitete kartice in kaj se takrat dogaja z zajemanjem po-

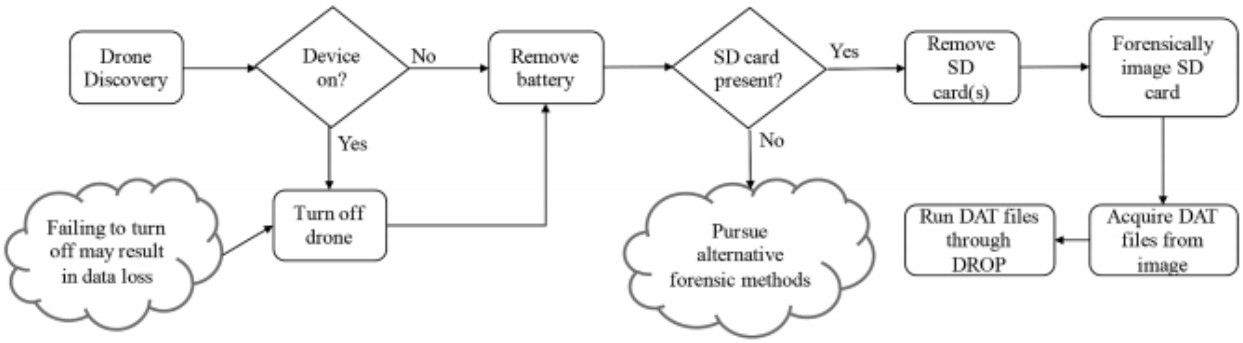


Figure 9: Diagram zajema podatkov iz DJI Phantom III drona

datkov. Datotečni sistem kartice je FAT32 in vsebuje približno 1182.13 MB prostora ter 39 DAT datotek. Izkazalo se je da se DAT datoteke izbrišejo od najstarejše naprej, ko je SD kartica zasedena. Predhodno jih tudi prepiše z ničlami in torej obnovitev ni možna.

Preizkus nad napravo je bilo tudi upravljanje brezpilotnega letala kjer so notranjo SD kartico odstranili. Pri samem letenju to ne predstavlja nobene ovire, je pa lahko kar velik problem za forenizčno raziskavo.

5. RELEVANTNI FORENZIčNI DOKAZI

Diagram poteka preiskave in zajema dokaznega gradiva je prikazana na sliki 8. Zajeti so bili pomebni forenzični podatki kot so GPS lokacija, WIFI povezave, informacije o uporabniku, časovni zapisi ipd., kot je povzeto v tabeli na sliki 9.

6. ZAKLJUčEK IN NADALJNO DELO

V obravnavanem članku je raziskava osredotočena zgolj na model drona DJI Phantom III. Ne povzema celotnega razumevanja forenzičnih raziskav nad široko paleto različnih proizvajalcev brezpilotnih letal, kljub temu pa predstavlja dobro izhodišče. Trenutno je nemogoče raziskati celoten trg. Podobne raziskave je potrebno še naprej izvajati in poskušati doseči tudi standardizacijo zapisa podatkov o aktivnostih dronov.

Precej dela je že bilo opravljenega nad varnostjo in zasebnostjo pri uporabi dronov ter postavljenih kar nekaj zakonom, ki omejujejo uporabo. Zelo malo pa nad forenzično analizo naprave kadar je z njo storjen zločin. Zmogljivost in uporaba dronov vedno bolj narašča in potrebno je pripraviti znanstveno metodologijo za ustrezno forenzično raziskavo.

V čanku je bilo prikazano kakšni podatki se lahko iz podobnih naprav pridobijo in so lahko ključni pri reševanju zločina. Vidna je tudi povezava med serijsko številko dronov in mobilne naprave, ki jo kontrolira. Še vedno je potrebno raziskti več informacij okoli strojno-programske opreme (angl. *firmware*) same naprave in kako ta deluje, saj zanašanje na notranji pomnilnik pogosto ne bo dovolj.

7. REFERENCES

[1] Dronelife. `https://dronelife.com/`. Dostopano: 2018-05-12.

[2] R. Brandom. Drone incidents 2016. `http://www.theverge.com/2016/3/25/11306850/faa-drone-airport-incidents`, 2016. Dostopano: 2018-05-12.

[3] D. R. Clark, C. Meffert, I. Baggili, and F. Breitinger. Drop (drone open source parser) your drone: Forensic analysis of the dji phantom iii. *Digital Investigation*, 22:S3 – S14, 2017.

[4] E. S. Michael S. Schmidt. Isis: Exploding drones. `https://www.nytimes.com/2016/10/12/world/middleeast/iraq-drones-isis.html`, 2016. Dostopano: 2018-05-12.

Finding	Description	Utilization
General findings		
External MicroSD card (64 GB)	Gimbal memory	Photo & video
Internal MicroSD card (4 GB)	Drone main board removable memory	Flight stats
Mobile device	Flight system feedback	Autopilot, pictures, GPS
External MicroSD		
Pictures/	/DJI_####.jpg	EXIF data (date/time, pitch, roll and yaw of Gimbal and aircraft)
Videos/	/DJI_####.m4v, /idx##	Metadata (file headers contained location of drone and GPS data)
Internal MicroSD		
DAT files/	/FLY###.DAT	Flight information (GPS, compass, battery, etc)
Nexus Tablet		
\apps\dji.pilot\db\dji.db	Pertinent flight and user information	No-fly zones, user email addresses, last known home point
\apps\dji.pilot\sp\dji.pilot.xml	device ID	device serial numbers, last flight location
\InternalStorage\DJI\dji.pilot\FlightRecord	DJIFlightRecord_YYYY-MM-DD_[HH-MM-SS].txt	Flight information (GPS, compass, battery, etc), user name, device serial numbers
\com.android.providers.settings\f\flattened-data	Credentials/SSID	WiFi credentials, user home address via access point identification
\Internalstorage\DJI\dji.pilot\CACHE_IMAGE	Images/EXIF data	Cached images from MicroSD on gimbal assembly

Figure 10: Povzetek forenzično relevantnih podatkov