



UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

UNIVERSITY OF LJUBLJANA
FACULTY OF COMPUTER AND INFORMATION SCIENCE

DIGITALNA FORENZIKA
DIGITAL FORENSICS

ZBORNIK
PROCEEDINGS

Seminarske naloge 2018/2019

Ljubljana, 2019



Zbornik / Proceedings

Digitalna forenzika / Digital forensics, Seminarske naloge 2018/2019

Editors: Andrej Brodnik, Nejc Nadižar, študenti

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko, Ljubljana, 2019.

® These proceedings are for internal purposes and under copyright of University of Ljubljana, Faculty of Computer and Information Science. Any redistribution of the contents in any form is prohibited. All rights reserved.

KAZALO / CONTENTS

1	Uvod / Introduction	1
2	Povzetki / Abstracts	2
2.1	In what legal context are digital forensics investigations set?	2
2.2	Standardiziran korpus za forenzičko SQLite podatkove baze	2
2.3	Forenzična preiskava Nintendo Wii: Prvi pogled	2
2.4	Styx: Countering Robust Memory Acquisition	2
2.5	Nadzorovani poskusi pri ponaredbi forenzičnih dokazov	3
2.6	Forenzička poškodovanih digitalnih naprav	3
2.7	Image tamper detection based on demosaicing artifacts	3
2.8	Using deep learning approach for malware classification	4
2.9	OpenForensics: A digital forensics GPU pattern matching approach for the 21st century	4
2.10	Forensic analysis of Mobile Applications on Android Devices with Fordroid	4
2.11	Rekonstrukcija pretočene video vsebine iz predpomnilnika spletnega brskalnika Chrome	5
2.12	The Growing Impact of IoT on Digital Forensics	5
2.13	Analysis of the BTRFS File System Using a Storage Pool Extension of The Sleuth Kit	5
2.14	Uporaba časovnega žiga v ext4 za skrivanje podatkov	6
3	Osnove in razno	7
	Basics and misc	
3.1	In what legal context are digital forensics investigations set?	8
3.2	Standardiziran korpus za forenzičko SQLite podatkove baze	10
3.3	Forenzična preiskava Nintendo Wii: Prvi pogled	14
4	Protiforenzika in pokvarjeno gradivo	19
	Antiforensics and damaged material	
4.1	Styx: Countering Robust Memory Acquisition	20
4.2	Nadzorovani poskusi pri ponaredbi forenzičnih dokazov	24
4.3	Damaged Device Forensics	29
4.4	Image tamper detection based on demosaicing artifacts	33
5	Metode	
	Methods	37
5.1	Using deep learning approach for malware classification	38
5.2	OpenForensics: A digital forensics GPU pattern matching approach for the 21st century	43
6	Omrežna in mobilna forenzika	
	Network and mobile forensics	49
6.1	Forensic Analysis of Mobile Applications on Android Devices with Fordroid	50
6.2	Rekonstrukcija pretočene video vsebine iz predpomnilnika spletnega brskalnika Chrome	55
6.3	The Growing Impact of IoT on Digital Forensics	62
7	Diskovna forenzika	
	Disc forensic	66
7.1	Analysis of the BTRFS File System Using a Storage Pool Extension of The Sleuth Kit	67
7.2	Uporaba časovnega žiga v ext4 za skrivanje podatkov	72



1 UVOD / INTRODUCTION

Digitalna forenzika je veja forenzične znanosti, ki zajema obnovo in preiskavo gradiva na jdenega v digitalnih napravah in je pogosto v povezavi z računalniškim kriminalom. Izraz digitalne forenzike je bil prvotno uporabljen kot sopomenka računalniške forenzike, 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 zgodnjih 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. Najpogosteje so podpirati ali ovreči hipotezo pred kazenskimi ali civilnimi sodišči. Kriminalni primeri vključujejo domnevno kršitev zakonov, ki jih določa zakonodaja, kot so na primer 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 2018/2019. 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 starih raziskovalnih področij: protiforenzika in pokvarjeno gradivo, metode, omrežna in mobilna forenzika, diskovna forenzika ter trije članki, ki pa niso spadali v nobeno od prej omenjenih kategorij.

Protiforenzika je pomemben del razvoja področja forenzike, saj dobro poznavanje le-te pripomore k nadaljnemu razvoju forenzičnih orodij in zaznavanju ponarejenih dokazov. Na področju protiforenzike in pokvarjenega gradiva, se seminarske naloge dotaknejo tematik forenzike pomnilnika, ponaredbe forenzičnih dokazov, pridobivanja podatkov iz fizično poškodovanih naprav in ponarejanja slik.

Sklop metod opisuje in predstavi nova orodja ter prijeme na področju digitalne forenzike. Seminarski nalogi pregledata področja uporabe strojnega učenja za klasifikacijo zlonamerne programske opreme in pregled paralelnega algoritma za ujemanje vzorcev na grafičnem procesorju.

Omrežna in mobilna forenzika sta trenutno še posebej pomembni, saj opažamo porast naprav, ki komunicirajo z omrežjem in svetovnim spletom. Seminarske naloge področja zajamejo forenzično analizo sistema Android z orodjem Fordroid, pregledajo možnost rekonstrukcije video vsebin iz predpomnilnika spletnega brskalnika Chrome in naredijo pregled vpliva interneta stvari (IoT) na področje digitalne forenzike.

Diskovna forenzika je kot ena pomembnejših vej vedno prisotna in se razvija na področju digitalne forenzike. Sklop obravnava rekonstrukcijo slik z uporabo SleuthKit na podatkovnem sistemu BTRFS in možnost skrivanja podatkov v časovnem žigu datotečnega sistema ext4.

Nenazadnje v sklopu razno seminarske naloge pregledajo, kako Nizozemska definira standarde sodnih izvedencev, standardiziran korpus za analizo podatkovne baze SQLite in forenzično preiskavo igralne konzole Wii.

Ta zbornik združuje vse končne seminarske naloge, ki so bile izdelane v študijskem letu 2018/2019. Namenjen je vsem, ki jih zanima področje digitalne forenzike ali pa zgolj eno ali več predstavljenih področij.

Ljubljana, 2019

Nejc Nadižar

2 POVZETKI / ABSTRACTS

2.1 In what legal context are digital forensics investigations set?

This article will discuss the article "Educating judges, prosecutors and lawyers in the use of digital forensic experts" that presents how the NRGD (Netherlands Register of Court Experts) recently defined legal standards about digital forensics investigation. These standards include demarcations between several subfields of digital forensics, a procedure of application for aspiring experts in these subfields and some prerequisites for the applicants to register as experts.

Ključne besede / Keywords: digital forensics, legal, standards, nrgd, investigation, experts

2.2 Standardiziran korpus za forenzično SQLite podatkove baze

Vedno več programov, kot so brskalniki ali aplikacije za pametne telefone, uporabljajo SQLite3 podatkovne baze za shranjevanje aplikacijskih podatkov. V mnogih primerih so taki podatki med forenzično preiskavo zelo dragoceni. Zato so bila razvita različna orodja, ki trdijo, da podpirajo strogo forenzično analizo datotek zbirke podatkov SQLite. Predstavljamo standardiziran korpus datotek SQLite, ki jih je mogoče uporabiti za vrednotenje in analizo metod in orodji. Korpus vsebuje podatkovne baze, ki uporabljajo posebne značilnosti SQLite datotek, ali pa vsebujejo potencialne pasti za zaznavanje napak v forenzičnih programih. Korpus uporabljamo za niz šestih razpoložljivih orodij, kjer ocenujemo njihove prednosti in slabosti. Pokazali smo, da nobeno od teh orodji ne more zanesljivo obdelati vseh robnih primerov formata SQLite3.

Ključne besede / Keywords: digitalna forenzika, forenzika podatkovne baze, sqlite3

2.3 Forenzična preiskava Nintendo Wii: Prvi pogled

Nove generacije igralnih konzol postajajo vse bolj zmogljive, podpirajo vse več storitev in zaradi tega hranijo vse več podatkov, ki jih je možno koristiti v forenzični preiskavi. Prav tako so postale tudi omrežne naprave z možnostmi brskanja po spletu, pošiljanja elektronskih sporočil in s tem postale bolj privlačne tudi za zlonamerno uporabo. Članek oriše postopke preiskave Nintendo Wii konzole, kjer preiskovalec lahko najde obilo informacij o zlonamerni uporabi. Tekom članka avtor opiše same značilnosti Nintendo Wii konzole in aplikacije, ki lahko preiskovalcem prispevajo k forenzični preiskavi ter se ob tem dotakne težav z ekstrakcijo le-teh. V pričajočem delu smo predstavili področje preiskave igralnih konzol, razširjeno povzeli značilnosti analize in zanimive kanale Nintendo Wii konzole ter njihove praktične implementacije.

Ključne besede / Keywords: forenzična analiza, digitalni dokazi, igralna konzola, nintendo wii

2.4 Styx: Countering Robust Memory Acquisition

Članek na kratko predstavi splošno področje pridobivanja spominskih slik digitalnih naprav z uporabo forenzičnih orodij, ter vsebino članka Ralph Palutke, Felix Freiling: Styx: Countering Robust Memory Acquisition (DFRWS EU 2018). Tekom procesa pridobivanja dokazov v postopku digitalne preiskave, preiskovalci pogosto uporabljajo raznovrstne forenzične tehnike ter forenzična programska orodja. Vse to z namenom pridobivanje natančne

slike glavnega pomnilnika obravnavanih digitalnih naprav. Pridobljene slike nato uporabljajo za ekstrakcijo digitalnih dokazov in identificiranje naprednih groženj. Med najbolj razširjeno in uporabljenou tehniko naprednega pridobivanja pomnilniških slik spada orodje pmem, ki sta ga razvila Johannes Stüttgen in Michael Cohen (Google). Izbran članek predstavlja t.i. proof-of-concept sistem imenovan Styx. Omenjen sistem z zakrivanjem sledi, preiskovalcem izniči učinkovitost uporabe predhodno omenjenega orodja pmem in preostalih klasičnih orodij, namenjenih pridobivanju pomnilniških slik. Implementacija sistema Styx sestoji iz naložljivega jedrnega modula, ki omogoča spodkopavanje 64-bitnih Linux sistemov z uporabo Intelove VT-x virtualizacije ter njene razširitve EPT.

Ključne besede / Keywords: robust memory acquisition, incriminating evidence, virtualization, linux, intel, styx

2.5 Nadzorovani poskusi pri ponaredbi forenzičnih dokazov

Število digitalnih dokazov narašča iz dneva v dan. Če je nekdaj veljalo, da so bili na sodišču pretežno fizični dokazi, danes velja obratno. Živimo namreč v digitalno dobi, kjer ima vsak posameznik dostop do več pametnih naprav. Digitalni dokazi prestavljajo določen izziv, saj jim ne moremo vedno zaupati. Te teme sta se dotaknila avtorja Felix Freiling in Leonhard Hösch, ki sta pri študentih digitalne forenzike naredila poskus ponaredbe digitalnih dokazov. V tem prispevku bomo njun poskus analizirali in predstavili ugotovitve. Gre za poskus manipulacije slike diska, pri čemer so morali študenti sliko diska spremeniti tako, kot da bi bil na njej opravljen dostop do spletne strani v preteklosti. Na koncu bomo tudi podali naše mnenje o preizkusu, kaj bi morda dodali ali spremenili.

Ključne besede / Keywords: antiforenzika, digitalna preiskava, ponarejanje dokazov, poskus

2.6 Forenzika poškodovanih digitalnih naprav

V zadnjem času je uporaba elektronskih naprav postala del našega vsakdana. V okviru digitalne preiskave lahko iz elektronskih naprav osumljenca pridobimo določene podatke, ki so lahko del dokaznega gradiva na sodišču. Zgodijo pa se primeri, ko je bila naprava ki jo dobimo v obravnavo podvržena določenim fizičnim poškodbam. Kljub poškodbam naprave želimo iz nje pridobiti uporabne podatke. Ker se poškodovane naprave ne znajdejo tako pogosto v forenzičnih preiskavah, še ni oblikovanega strokovnega znanja, kako obravnavati poškodbe naprave. Prav tako na to temo še ni bilo naslovljeno veliko raziskav. V članku bomo opisali glavne vrste poškodb na digitalnih napravah, ter kako postopati v primeru obravnavne poškodovane naprave. Pregledali bomo tudi načine pridobitve podatkov iz poškodovanih mobilnih naprav, ter možnosti pridobitve podatkov iz poškodovanih pomnilniških medijev.

Ključne besede / Keywords: poškodovane naprave, tipi poškodb, forenzika pomnilnika, pridobivanje podatkov

2.7 Image tamper detection based on demosaicing artifacts

This paper reviews and summarizes research done on image tampering detection using a color filter array demosaicing technique. The algorithm and it's features are explained and implemented in MATLAB. The algorithm is tested on sample data and results are discussed.

Ključne besede / Keywords: image tampering, digital forensics, color filter array

2.8 Using deep learning approach for malware classification

In this paper we present work done by researchers regarding malware classification using deep neural networks. Their approach used CNN in combination with LSTM neural network to statically analyze malware binary code, which was beforehand transformed into gray-scale picture. Their work improves upon existing techniques and drastically decreases entry level for non expert personnel, which can effectively use proposed deep neural network model. It achieved final accuracy score of 98.8

Ključne besede / Keywords: machine learning, malware, deep neural networks, long-short term memory

2.9 OpenForensics: A digital forensics GPU pattern matching approach for the 21st century

It happens that filesystems become corrupted. Sometimes, it may be done intentionally by a person that committed a crime. In such a case, the process called File carving is used to recover files. This process is really useful during a digital investigation. In the paper for which we are giving a survey, the authors provided a new approach for recovering files. They came with a framework called *OpenForensics* which contains a parallel algorithm that process pattern matching on GPU. Their approach is based on *Parallel Failureless Aho-Corasick* algorithm (PFAC) which search for multiple patterns simultaneously. They also implemented the Open-Forensics framework and compared it to commonly used digital forensics tools (Foremost, Recover My Files). Results show that their framework outperforms both tools. Further more, their algorithm performs the pattern matching almost as fast as the read speed of the storage device is.

Ključne besede / Keywords: digital forensics, processing model, pattern matching, asynchronous processing, gpu, gpgpu

2.10 Forensic analysis of Mobile Applications on Android Devices with Fordroid

Over the course of the past two decades, phones stopped just being phones. Nowadays, phones are not only used to send voice and text messages but also to browse Internet and send emails, take photos, listen to music, get directions to anywhere you need to go, play video games, pay groceries (and bills), and more. Mobile devices have become an essential part of our lives, but also have become a part in criminal activities. As the involvement in these criminal activities increases, the need to rapidly tackle these activities increases. Digital forensics can be defined as the process of collecting, examining, analyzing and reporting of digital evidence without any damage. Digital forensics requires a detailed examination of devices such as computers, mobile phones, sd cards, tablets that contain digital evidence. However, traditional forensic approaches, which are based on the manual investigation are not scalable to numerous mobile applications. So this paper is presenting a *fully automated* tool named **Fordroid** for the forensic analysis of mobile applications on Android. This tool conducts inter-component statistic analysis on Android APK's and builds control flow and data dependency graphs. Also identifies what and where the information is written in local storage, and how the information is stored by parsing

SQL commands. **Fordroid** is tested on 100 randomly selected Android applications and from the analyses, it can be seen that the success rate on locating where the information was written is 98

Ključne besede / Keywords: digital forensics, static analysis, android, fordroid, local storage

2.11 Rekonstrukcija pretočene video vsebine iz predpomnilnika spletnega brskalnika Chrome

Zmožnost ogleda video vsebin preko spletja je danes ena izmed najbolj zaželenih aktivnosti vsakodnevnega uporabnika svetovnega spletja. Današnji brskalniki omogočajo, da video vsebine za ogled ne potrebujemo v celoti prenašati na napravo, ampak so jo sposobni pretakati (angl. streaming), kar pomeni, da lahko s predvajanjem video vsebin pričnemo takoj, le-ta pa se v majhnih koščkih prenaša na napravo in se shranjuje v predpomnilnik (angl. cache) brskalnika. Z vidika forenzične analize je to zanimiv problem, ker predvajana in potencialno ogledana video vsebina ni bila nikoli neposredno prenešena na napravo. V datotečnem sistemu se datoteka z video vsebino ni nikoli nahajala v celoti, temveč je video vsebino potrebno rekonstruirati iz večih datotek predpomnilnika brskalnika, v katerem je bila predvajana.

Ključne besede / Keywords: predpomnjene, rekonstrukcija, digitalna forenzika, google chrome, youtube, facebook live, twitch

2.12 The Growing Impact of IoT on Digital Forensics

Internet of Things (IoT) strives to create a highly heterogeneous network of interconnected nodes that autonomously communicate with each other. Although the nature of devices can be directly related with conventional digital forensics, IoT brings a number of challenges that add additional complexity to the existing digital forensic investigations. In this paper we overview the field of IoT and its difference from traditional (conventional) digital forensics. We also break down different types of devices, define which devices fall into each category, their hardware composition and what kind of data we can gather from them. Finally we look at the number of challenges that field brings or changes and provide brief overview for each one.

Ključne besede / Keywords: digital forensics, internet of things, iot, security

2.13 Analysis of the BTRFS File System Using a Storage Pool Extension of The Sleuth Kit

BTRFS is a Linux based file system that almost exclusively uses B-trees to store its structure according to the copy on write principle. The file system enables us to use subvolumes, which can be viewed as mountable directories in order to organize our directory structure and roll back to previous snapshots of the file system. Unlike older file systems, BTRFS was designed for pooled storage.

The Sleuth Kit is a forensic toolkit that allows forensic analysis regardless of the file system. Hilgert et al. implemented an extension for TSK with BTRFS support. Their implementation provides all essential commands to perform forensic analysis of the BTRFS file system even when disks are missing from the pool.

We attempt to replicate Hilgert et al. results of JPG image data extraction using their

BTRFS extension of The Sleuth Kit while one out of two BTRFS disks are missing. We are able to partially recover text files, but images are either unreadable or showed no discernible content.

Ključne besede / Keywords: digital forensics, the sleuth kit, btrfs, pooled storage

2.14 Uporaba časovnega žiga v ext4 za skrivanje podatkov

Ext4 je priljubljen datotečni sistem, ki ga uporablja Android in številne distribucije Linuxa. Ta dokument analizira izvedljivost uporabe časovnih žigov datotečnega sistema ext4 za skrivanje podatkov. Najprej preučimo uporabo, strukturo in zmogljivost razpoložljivih časovnih žigov. Rezultati razkrijejo, da je del nanosekundnih časovnih žigov ext4 mogoče uporabiti za izgradnjo steganografskega sistema. Poleg tega opišemo ext4 anti-forenzično tehniko, ki omogoča tajnost skritih podatkov in enostavno uporabo v širokem spektru scenarijev. Opisan je niz zahtev (npr. nerazpoznavnost rednih in nepooblaščenih časovnih žigov) in koncept, ki lahko prikrije poljubne podatke v časovnih žigih datotečnega sistema.

Ključne besede / Keywords: digitalna forenzika, skrivanje podatkov, ext4, nanosekundni časovni žigi, steganografija, anti-forenzična tehnika, forenzična tehnika, forenzična analiza, forenzični sistemi



3 OSNOVE IN RAZNO BASICS AND MISC

In what legal context are digital forensics investigations set ?

Brieuc Vably
bv4647@student.uni-lj.si

ABSTRACT

This article will discuss the article "Educating judges, prosecutors and lawyers in the use of digital forensic experts" that presents how the NRGD (Netherlands Register of Court Experts) recently defined legal standards about digital forensics investigation. These standards include demarcations between several subfields of digital forensics, a procedure of application for aspiring experts in these subfields and some prerequisites for the applicants to register as experts.

Keywords

Digital forensics, legal, standards, NRGD, investigation, experts

1. INTRODUCTION

Recently the amount and the complexity of viruses have exponentially increased, especially the ransomware cases, subsequently, the volume of digital evidence to process is increasing too.

To face this problem, the NRGD (Netherlands Register Court of Experts) has taken some measures. This first implied setting standards about digital forensics and define processes of application for experts, then defining what exactly are digital forensics and theirs subfields. Finally the NRGD set some prerequisites for the experts to be accepted as court experts.

All these measures now rule how digital evidence and experts are processed and used by judges, lawyers and prosecutors in criminal investigations.

2. SETTING STANDARDS

2.1 Subfields

In 2015, the NRGD standards on digital forensics and application instructions were published on their website. First of all the very wide field of digital forensics had to be subdivided in many smaller fields. Six generic different subfields

were defined, if these subfields were more defined and specific, they risked to be outdated way quicker due to the quick progress of technologies.

2.2 Consultation

After defining these subfields, a procedure had to be set for submitting new standards in these different fields. A first set of standards was published on the website of the NRGD by the Advisory Committee for Standards, asking for digital forensics experts to comment and evaluate these new standards. A new draft of standards made from the reviews of the experts is then published and taken to the Board. These new standards are then adopted.

2.3 Assessment of experts

Some aspiring experts in digital forensics then are assessed by a committee composed of experts from different countries (ACA) on these standards. The applicants only come from the Netherlands for the moment but also foreigners can become court experts. This procedure of application will allow the NRGD to have enough experts in all the sub fields in a few years.

2.4 Prerequisites to be an expert

An expert must be able to elaborate on a strategy of investigation and do some relevant research, to collect, evaluate and document some data in a forensic context, and, to investigate in a forensic context.

The NRGD also asks experts to have at least a 3 years working experience in the field of information technologies at the level of a master's degree or 5 years in the case of a bachelor's degree. The candidate also has to have reported at least 5 cases in the last 5 years (due to the quick development of IT).

The expert also has to have knowledge about the dutch criminal law, and in the case of foreign aspiring experts, they will have to follow an introduction course to the dutch criminal law authorized by the NRGD.

3. DEFINITION OF DIGITAL FORENSICS

3.1 Tasks or activities

First digital forensics investigations are divided in 3 different main tasks :

1. data collection : which involves the collection and copy of data without causing any damage to them
2. data examination : is about investigating a data source to find files etc
3. data analysis : relates to the processing of data that has been found after the previous two tasks

You have to be able to lead all of these 3 tasks to be called a digital forensics expert.

3.2 Subfields definition

As mentioned in the paragraph 2.1 6, the six following subfields of digital forensics were defined :

1. Computer forensics : refers to all the methods, skills and techniques used to gather and analyze data from storage devices (for example : browser history, files and their timestamps and ownerships)
2. Software forensics : mainly relates to analyzing source codes to investigate on the ownership of a software (this can lead the expert to know if a software licence has been stolen or ripped)
3. Database forensics : refers to investigate about databases and their content (files, when they were last read or written etc), this requires a lot of knowledge about about database languages (SQL for exemple) and about the different database vendors which all have some particularities.
4. Multimedia forensics : focuses on analysis and recovery of media files such as audio records, movies and pictures and the analysis of their metadata (for instance EXIF data on a picture to get some GPS coordinates)
5. Device forensics : refers to the analysis of different devices such as mobile phones, cameras, storage devices, hard drives, tablets etc. This subfield is separated from the others even though all these devices are basically computers because most data on the devices are embedded and closed. That means that it requires a lot of knowledge to collect and process them.
6. Network forensics : focuses on data related to networks like telecom, internet, wireless and clouds.

3.3 Limitations of these subfields

Although these 6 subfields were defined by the NRGD, we've seen that they are still very wide and that they include a lot of different technologies. We also know that new technologies always appear or evolve very quickly, so these subfields are also bound to change rapidly. But this "problem" can easily be fixed thanks to the measures taken by the NRGD (paragraph 2.2), which allows experts to suggest new standards to be set. Also digital forensics experts have to be educated to all these subfields because in many cases, digital crimes involve a lot of different ones.

4. STRENGTH AND WAYS OF IMPROVEMENTS

4.1 Strengths

4.1.1 *Adaptative*

The first strength of the legal system that was reached by the NRGD with setting all these standards is that it can be easily adapted and modified. It is a really important aspect especially in the field of digital forensics, which is constantly evolving and expanding.

The procedure that was created in order to submit new standards was really necessary in this field, and it has been well made.

4.1.2 *Collaborative*

Secondly, the fact that digital forensics experts can give their word on new suggestions of standards in order to modify them is a positive point. By increasing the number of opinions and review we can avoid setting wrong standards or making huge mistakes.

Also the NRGD allows people from different countries to become court experts in the Netherlands, they even offer them education to the Netherlands' criminal legislation. It is also interesting to get point of views from people of other nationalities, that may have a different legislation about digital forensics.

4.2 Ways of improvement

4.2.1 *International cooperation*

Rather than getting foreign candidates to study the legislation of the Netherlands, Maybe getting court experts from european countries or cooperating with other institutions would be a better idea. It would allow the NRGD to avoid dispensing education about the dutch law to candidates.

The Netherlands should also take examples of other countries that tried to install legal standards because they might have already encountered some failures that could be avoided by taking some early measures.

5. CONCLUSIONS

During this paper we have studied the way that the NRGD have created standards to legally endorse digital forensics investigation. They have made a very strong system that can allow to easily adapt and evolve the standards along with the new and fast changing technologies. However some points are to improve and even though the NRGD has experience with a lot of forensics domains, we can never know what to expect in a domain like digital forensics.

6. REFERENCES

Hans Henseler, Sophie Van Loenhout: Educating Judges, Prosecutors and Lawyers in the Use of Digital Forensic Experts (DFRWS EU 2018) <https://www.sciencedirect.com/science/article>

Standardiziran korpus za forenziko SQLite podatkovne baze

Sebastjan Kozoglav

Fakulteta za Računalništvo in Informatiko

Večna pot 113

Ljubljana, Slovenija

sk5429@student.uni-lj.si

ABSTRACT

Vedno več programov, kot so brskalniki ali aplikacije za pametne telefone, uporabljajo SQLite3 podatkovne baze za shranjevanje aplikacijskih podatkov. V mnogih primerih so taki podatki med forenzično preiskavo zelo dragoceni. Zato so bila razvita različna orodja, ki trdijo, da podpirajo strogo forenzično analizo datotek zbirke podatkov SQLite. Predstavljam standardiziran korpus¹ datotek SQLite, ki jih je mogoče uporabiti za vrednotenje in analizo metod in orodji. Korpus vsebuje podatkovne baze, ki uporabljajo posebne značilnosti SQLite datotek, ali pa vsebujejo potencialne pasti za zaznavanje napak v forenzičnih programih. Korpus uporabljamo za niz šestih razpoložljivih orodij, kjer ocenjujemo njihove prednosti in slabosti. Pokazali smo, da nobeno od teh orodji ne more zanesljivo obdelati vseh robnih primerov formata SQLite3.

Keywords

digitalna forenzika, forenzika podatkovne baze, SQLite3

1. UVOD

SQLite se je razvil v eno najbolj uporabljenih sistemov za upravljanje podatkovnih baz (DBMS) na svetu [2]. Prvotno usmerjen na vgrajene naprave, je njena samostojna DBMS izvedba s celovito podporo za SQL predstavlja popularen sistem shranjevanja aplikacij za sporočanje in brskanje po spletu, predvsem iz mobilnih naprav. Pogosta uporaba SQLite je vodila do položaja, ko digitalne raziskave redno zahtevajo forenzično analizo podatkov, shranjenih v podatkovnih bazah SQLite.

1.1 O pomenu forenzičnih korpusov

Kot na mnogih drugih področjih forenzičnega računalništva, so prva orodja za forenzično analizo datotek SQLite ustvarili praktiki v odgovor na nujne potrebe. Proizvajalci komercialnih orodij so prispevali svoj delež odgovorov na neizbežne

¹nabor pisnih in govorjenih besedil

tržne potrebe. Malo pa je znanega o ravni zanesljivosti in pregledu, s katerim lahko ta orodja analizirajo splošne datoteke SQLite. Podrobno poznavanje prednosti in slabosti orodji je konec koncov potrebno v forenzični analizi.

Brez standardiziranih korpusov se prispevki pogosto zanašajo na osebne in dokaj majhne sklope podatkov, ki se včasih ustvarijo le za zelo poseben namen. Takšni nizi podatkov običajno niso na voljo javnosti. Drugi raziskovalci ne morejo niti reproducirati, preverjati niti graditi na predstavljenih testnih primerih in rezultatih. Ne moremo objektivno primerjati prednosti in slabosti različnih orodij med seboj brez standardiziranega in javno dostopnega testnega sklopa podatkov. Na kratko raziskovalna prizadevanja in rezultati, ki se izvajajo na posameznih nizih podatkov, niso primerljivi in zato manj koristni. Takšna prizadevanja imajo omejen vpliv in so v bistvu izgubljena za prihodnje raziskave.

Bolj znanstveni pristop je omogočiti javnosti dostop do podatkov. Na ta način jih lahko uporabljajo različne osebe za različne namene skozi čas. Z javno dostopnim testnim skupkom podatkov lahko neposredno primerjamo nove pristope in algoritme za forenzično analizo z obstoječimi orodji. Možne izboljšave, kot je manjša poraba virov ali izboljšana učinkovitost, lahko izmerimo in podamo na voljo vsem. Sprejem orodja s strani uporabnikov ne temelji več na ugledu njegovih ustvarjalcev, temveč se lahko izpelje iz javno dostopnih in ponovljivih rezultatov preskusov. Uporabniki lahko povečajo zaupanje v orodje in pregledajo njegovo kakovost. Na ta način lahko javna revizija programske opreme naredi orodja bolj robustna in zanesljiva. To je tisto, kar moramo zahtevati od orodja, zlasti kadar se uporablja v forenzičnih preiskavah.

2. UVOD V SQLITE FORENZIČNI KORPUS

Zaradi pomembnosti SQLite podatkovnih baz v digitalni forenziki, menimo, da je vredno temu področju posvetiti ločen korpus in s tem omogočiti, da se osredotočimo na tehnične podrobnosti osnovnega formata datotek [1]. V nadaljevanju bomo predstavili rezultate raziskave [5].

Korpus za forenziko SQLite podatkovne baze, se osredotoča na strukture formata datotek, zato vsebuje posebej izdelane datoteke baz podatkov, ki so bile ustvarjene z uporabo SQLite DBMS.

Korpus vsebuje izključno testne podatke. Podatkovne baze so bile sintetično oblikovane za opredelitev različnih test-

Figure 1: Kategorije in mape korpusa

Kategorija	Mapa	Tema
Ključne besede in identifikatorji	01	Neneavadna imena tabel
	02	Enkapsulacija definicije stolpcev
	03	SQL ključne besede in omejitve
Kodiranje	04	Nabor znakov UTF
Elementi podatkovnih baz	05	Prožilci, pogledi in indeksi
	06	Navidezne in začasne tabele
Struktura dreves in strani	07	Razdrobljene vsebine
	08	Rezervirani bajti na stran
	09	Stran s kazalcem
Izbrisana in prepisana vsebina	0A	Izbrisane tabele
	0B	Prepisane tabele
	0C	Izbrisani zapisi
	0D	Prepisani zapisi
	0E	Izbrisane strani

nih scenarijev. Ti temeljijo na različnih vsebinah v podatkovnih bazah ali na različnih vrednostih notranjih struktur, ali oboje. Korpus, ki vsebuje testne podatke, ima več prednosti, ki jih je mogoče uporabiti za testiranje in razvoj orodja. Ena od prednosti je obstoječe znanje o resnici, ki jo dokumentiramo v metapodatkih, ki spremljajo korpus. To omogoča merjenje zmogljivosti in kakovosti preizkušenih orodij ter medsebojne primerjave rezultatov testov različnih orodij. Druga prednost je, da distribucija korpusa ni omejena na noben način. Lahko je prosto dostopen in deljiv med ljudmi po celi svetu.

SQLite forenzični korpus obsega skupaj 77 podatkovnih baz. Vsaka datoteka z zbirko podatkov je bila posebej oblikovana in ima vsaj eno posebnost v njeni vsebini ali notranji strukture. Skupna obdelava podatkovnih zbirk v korpusu pomeni testiranje orodja proti številnim različnim scenarijem, ki jih je treba preveriti.

Zaradi lažjega razumevanja in opisovanja posebnosti, ki so prisotne v korpusu, zbirke podatkov razvrstimo v 14 map, dodeljenim petim kategorijam, prikazano na sliki 1. Vsaka mapa vsebuje eno ali več datotek baze podatkov, ki so del ene kategorije. Dodatno so na voljo ločene datoteke z metapodatki o vsaki bazi podatkov.

3. PODATKOVNE BAZE V SQLITE FORENZNEM KORPUSU

Celotna struktura baze podatkov je definirana v shemi podatkovne baze. Shema je podana z nizi ukazov SQL, ki opisujejo vsak posamezen element. Splošni način shranjevanja vnosa ali vrstice v podatkovni bazi SQLite lahko primerjamo s shranjevanjem datoteke v datotečnem sistemu. Prostor za shranjevanje je razdeljen na zaporedne bloke fiksne velikosti. V datotečnih sistemih se sklicujemo na te bloke kot na gruče, medtem ko se v podatkovnih bazah SQLite osnovni bloki imenujejo strani. Za shranjevanje katere koli vsebine, ki jo predstavlja določena količina bajtov, je dodeljenih toliko blokov, kot je potrebno. Navadno se kazalci uporabljajo za referenčne bloke in za dodelitev določenih blokov določenemu subjektu, kot je zapis baze podatkov.

Pri gradnji korpusa je bilo ustvarjenih več posebnosti z vnašanjem nenavadnih vrednosti za različne strukture znotraj baze podatkov. Vse datoteke še vedno v celoti ustrezajo osnovnemu formatu datoteke, zato so baze podatkov v veljavnem in skladnem stanju. Namen posamezne datoteke je prikazati želeni scenarij.

Privzeta velikost strani je 4096 bajtov, privzeto kodiranje baze podatkov pa je UTF-8. V nadaljevanju bomo opisali vseh 77 datotek baze podatkov v korpusu, razvrščenih po kategorijah, kot je prikazano na sliki 1.

3.1 Ključne besede in identifikatorji

Značilnosti podatkovnih baz v tej kategoriji so neneavadna imena tabel, enkapsulacija definicije stolpcev in določene ključne besede SQL. Testiranje v zvezi s to kategorijo lahko kaže, ali orodje pravilno obravnava stavke SQL, ki jih vsebuje baza podatkov.

3.2 Kodiranje

Baze podatkov v tej kategoriji imajo različna besedilna kodiranja. SQLite podpira tri tipe kodiranja: UTF8, UTF16-le in UTF16-be. Te podatkovne baze vsebujejo tudi nelatinične znake. Preizkuševanje v zvezi s to kategorijo lahko pokaže, ali orodje pravilno obravnava vsebino baze podatkov z različnimi kodiranjami.

3.3 Elementi podatkovnih baz

Podatkovne baze v tej kategoriji imajo različne vrste elementov. Medtem ko večina scenarijev v korpusu temelji na tabelah podatkovne baze (saj so ti najpomembnejši), te baze podatkov vsebujejo tudi navidezne in začasne tabele, indekse, prožilce in poglede. Ena podatkovna zbirka z začasno tabelo je morala biti izrezana iz RAMa, da bi se ohranila. Preizkušanje v tej kategoriji lahko pokaže, ali orodje pravilno obdeluje elemente baze podatkov, ki niso običajne tabele [4, 3].

3.4 Strukture dreves in strani

Podatkovne baze v tej kategoriji vsebujejo različne scenarije glede notranjega drevesa in postavitve strani baz podatkov. Kadar mora SQLite shraniti vsebino določene dolžine, ki ne ustreza eni strani, se zapis razdeli in deli se shranijo na eni ali več prelivnih strani. Preizkušanje v tej kategoriji lahko pokaže, ali orodje pravilno obravnava strukture drevesa in strani.

3.5 Izbrisana in prepisana vsebina

Podatkovne baze v tej kategoriji vsebujejo artefakte izbrisanih podatkov. Scenariji segajo od izbrisanih in prepisanih tabel nad izbrisane in delno prepisane zapise do izbrisanih strani. Vse nastanejo iz različnih zaporedij stavkov SQL CREATE, INSERT, DELETE in DROP. Kadarkoli je zaželeno, se nekaj novih vsebin vstavi po izbrisu drugih, da se prepričejo sledi predhodno izbrisane vsebine. Za nove vstavljenе vnose so bili nekateri IDji, ki enolično identificirajo vnos, morda uskljeni z IDji predhodno izbrisanih vnosov. Preizkuševanje v zvezi s to kategorijo lahko pokaže, ali orodje lahko pravilno obnovi izbrisane vsebine, če niso prepisane.

4. EVALVACIJA

Ko smo ustvarili korpus, smo pogledali, kako se orodja obnašajo v različnih scenarijih. V zvezi s forenzično analizo obstajata na splošno dve vrsti programske opreme: orodja, ki ne obnavljajo izbrisanih artefaktov in orodja, ki to storijo. Po eni strani, prvi niz orodji preprosto izvleče podatke logično prisotne v podatkovni bazi. Pri vrednotenju takšnih orodij, ki delujejo kot brskalnik ali pregledovalnik baz podatkov, lahko izmerimo količino vsebin, ki se razlagajo in predstavijo na pravilen način. Po drugi strani pa drugi sklop orodji trdi, da dodatno obnavlja artefakte predhodno izbrisanih vsebin, če so prisotne. Takšna orodja običajno izrezujejo vsebine iz nedodeljenih področij znotraj datoteke.

Izbrali smo šest različnih orodij in jih evaluirali z novim korpusom. Nekatera orodja so prosto dostopna, nekatera pa so komercialna. Nekatere je treba obravnavati kot pregledovalce podatkovnih zbirk, medtem ko se druge promovirajo, da podpirajo forenzično analizo datotek zbirk podatkov SQLite, vključno z obnovitvijo izbrisanih vsebin. Izbrana orodja so: *Undark*, *SQLite Deleted Records Parser*, *SQLite Doctor*, *Stellar Phoenix Repair for SQLite*, *SQLite Database Recovery in Forensic Browser for SQLite*.

V nadaljevanju bomo opisali rezultate glede na posamezno kategorijo.

4.1 Ključne besede in identifikatorji

Za prvo mapo, kjer ime tabele vsebuje samo značke presledkov, večina programov ne prikaže pravilno. Vsi programi, ki temeljijo na oknu, vrnejo napačno ime tabele. Ime tabele z odprtimi oklepaji povzroči napako v treh in tabela z zaprtimi oklepaji v dveh programih. Tриje programi ne morejo obdelati praznega imena tabele, medtem ko ima *Forensic Browser* težave z vsemi bazami podatkov v tej kategoriji. Mnogi programi vračajo napake pri analizi podatkovnih baz, ki vsebujejo enkapsulirane znake v ključnih besedah in identifikatorjih. *SQLite Doctor* ne uspe samo pri nenavadnih imenih stolpcev, vendar nima nobenih težav z enkapsuliranimi definicijami. *Database Recovery* pravilno reši dva scenarija izmed sedmih, *Forensic Browser* pa samo enega. Vse druge podatkovne baze se s temi orodji nepravilno obravnavajo. *Phoenix Repair* v celoti ne uspe za definicije stolpcev. V zvezi s posebej oblikovanimi ključnimi besedami in omejitvami SQL lahko sklepamo, da nobeno orodje ne obdeluje podatkovne baze brez ukazne vrstice. Vendar pa večina drugih scenarijev ne povzroča resnih težav.

4.2 Kodiranje

Forensic Browser ne uspe v podatkovnih bazah kodiranih v UTF-16be. *Phoenix Repair* ne more obdelati nobenih kodiranih baz podatkov UTF16. Prav tako ne uspe pri nela-tiničnih znakih. *Undark* lahko natisne samo ASCII znake, medtem, ko *SQLite Doctor* vedno pretvori v UTF8.

4.3 Elementi podatkovnih baz

Pri analiziranju prožilca, pogleda in indeksa, *Database Recovery* ne prikaže indeksa. Vsi drugi scenariji so pravilno obdelani z ustreznimi orodji. Vendar noben od programov ne more upravljati podatkovnih baz z uporabo R* tree modula, razen *SQLite Doctor*. Slednji kopira vsebine iz stare podatkovne baze v novo, kar ima za posledico veljavno bazo

podatkov, vendar se med analizo prikaže veliko sporočil o napakah. Vsebino podatkovne baze z uporabo razširitve FTS lahko prikaže vsak program.

4.4 Strukture dreves in strani

Pri nekaterih zapisih smo zaznali le manjše napake. *Phoenix Repair* ima največ težav, ker ne more prikazati celotnega zapisa v tri od štirih scenarijev. Glede rezerviranega prostora na koncu vsake strani je edini program, ki omogoča uporabniku, da najde skrita sporočila *SQLite Parser*. Oba skrita besedila je mogoče najti v izhodu programa. Odločilen je rezultat, ki ga je vrnil *SQLite Doctor*: izhodna baza podatkov, ki izhaja iz analize, se zdi, da je skladna s standardno predlogo, s privzetim kodiranjem in brez rezerviranega prostora. To pomeni, da *SQLite Doctor* v tem primeru uniči nekaj sledi. Datoteka zbirke podatkov, ki vsebuje stran kazalca, je pravilno obdelana z vsemi orodji, razen z orodjem *Database Recovery*. Slednji je prikazal različne količine vnosov med večkratnim poganjanjem, za kar ni očitne razlage.

4.5 Izbrisana in prepisana vsebina

Oblikovani scenariji glede izbrisanih tabel so zelo različni. Na primer, v nekaterih primerih so bili vsi zapisi izbrisani, preden je bila izbrisana tabela, medtem ko so bile v drugih primerih tabele izbrisane brez brisanja zapisov. Vrednotenje kaže različne rezultate za te primere: Brez brisanja zapisov *Undark* obnovi vsak zapis. Ob izbrisu *Undark* obnovi približno polovico zapisov. *SQLite Parser* se obnaša obratno, saj lahko povrne besedilna polja iz izbrisanih tabel z izbrisanimi zapisi, vendar nobenega od zapisov neizbrisanih. Niti *Phoenix Repair* niti *Forensic Browser* ne moreta obnoviti enega samega zapisa.

V primeru prepisanih tabel je globlja analiza baz podatkov pokazala, da obnovljenih izbrisanih zapisov ni mogoče obnoviti, ker se zdi, da so ustreznna pomnilniška področja strani očiščena z null bajti ob ponovni uporabi, tudi če je secure delete nastavitev deaktivirana. Zato nobeden obnovitvenih programov ne more obnoviti enega samega zapisa. *SQLite Doctor*, *Phoenix Repair* in *Database Recovery* lahko natisnejo logično obstoječe zapisne, tudi če so v bazi podatkov izbrisani zapisni. Vendar pa *Undark* kaže nepričakovano vedenje, ker se v nekaterih primerih ne prikažejo niti obstoječi niti izbrisani zapisni, če so v bazi podatkov izbrisani zapisni. *SQLite Parser* je uspešno obnovil vsako izbrisano besedilno polje. *Forensic Browser* lahko včasih obnovi vsak zapis in včasih le nekaj zapisov. S tem orodjem se lahko obnovljeni zapisni dodelijo napačni tabeli. Če je način obnovitve omogočen, se ne prikažejo niti obstoječi niti izbrisani zapisni, če vsebujejo številke s plavajočo vejico. Te se prikažejo samo s strani *Forensic Browser-ja*, ko je v načinu neobnovitve. Tako nobeno orodje ne more obnoviti številki s plavajočo vejico, medtem ko samo *Forensic Browser* pravilno obnovi vrednosti celih števil in samo *SQLite Parser* omogoča obnovitev vseh besedilnih polj.

Pri obnovitvi prepisanih zapisov, je ena od glavnih razlik nezmožnost *Forensic Browser-ja*, da prikaže prepisane zapisne. *SQLite Parser* prikaže vsaj nekaj besedil, vsebovanih v prepisanih zapisih. Prav tako se zdi, da obnovitev zapisov z brisanimi strani ni tako zanemarljiva. Iz prve baze podatkov so tri orodja, ki ponujajo odbrisanje (*Undark*, *SQLite*

Parser, Forensic Browser). Iz druge baze podatkov nobeden od programov ne more popolnoma obnoviti zapisa, nekateri le v delih.

5. ZAKLJUČEK

V tem članku, predstavljamo - v našem znanju - prvo standardizirano korpusno ciljanje baz podatkov SQLite: SQLite Forensic Corpus. Lahko se uporabi za testiranje in validacijo orodja ter za razvoj in primerjavo novih algoritmov ter orodji. Vsebuje 77 datotek baze podatkov, ki v celoti ustreza formatu datoteke SQLite3. Petdeset datotek baz podatkov se osredotoči na izdelane posebnosti stavkov SQL in notranjih struktur v formatu datoteke, medtem ko sedemdvajset podatkovnih baz vsebuje izrecno izbrisane artefakte, ki jih je mogoče obnoviti.

Na novo oblikovan korpus je bil preizkušen s strani šestih različnih programov, ki so bili usmerjeni na analizo podatkovne baze SQLite. Polovico orodji je potrebno obravnavati kot gledalce baz podatkov, drugo polovico pa spodbujamo, da bi lahko obnovili podatke, ki so bili prej izbrisani. Rezultati jasno kažejo, da ni popolnega orodja za analizo SQLite podatkovne baze, saj se vsi borijo z različnimi posebnostmi. Zlasti ne izbris vnosov, ki vsebujejo številske vrednosti, pa naj bo to celo število, ali število s plavajočo vejico, povzroča resne težave orodjem za obnovitev. Naši rezultati se lahko uporabijo za pomoč forenzičnim preiskovalcem pri izbiri orodja z najmanj pomanjkljivostmi za določeno nalogu analize.

Iz rezultatov svojega dela povzemamo naslednje ugotovitve. Te bi bilo treba razumeti kot osnovne zahteve, ki jih mora forenzično orodje izpolnjevati in jih je mogoče obravnavati kot smernico v prihodnjem razvoju kakršne koli programske opreme za analizo SQLite:

1. Sledi iz dokazov naj se ne uničijo, ko se pretvorijo ali prenesajo na izhod podatkov forenzične analize.
2. Napačna analiza v delih dokazov ne povzroči odprave ali opustitve drugih delov, kot so tisti, ki še niso bili analizirani.
3. Analiza obstoječih, logično predstavljenih dokazov, če je podprta, se ne sme zmanjšati z aktiviranjem ali uporabo funkcionalnosti za obnovitev podatkov.

6. REFERENCES

- [1] Sqlite online documentation. file format for sqlite databases, Citirano Maj 2019.
- [2] Sqlite online documentation. most widely deployed sql database engine, Citirano Maj 2019.
- [3] Sqlite online documentation. sqlite fts3 and fts4 extensions, Citirano Maj 2019.
- [4] Sqlite online documentation. the sqlite r*tree module, Citirano Maj 2019.
- [5] S. Nemetz, S. Schmitt, and F. Freiling. A standardized corpus for sqlite database forensics. *Digital Investigation*, 24:S121–S130, 2018.

Forenzična preiskava Nintendo Wii: Prvi pogled

[Razširjen povzetek]

Andreja Kovačič Fakulteta za računalništvo in informatiko Večna pot 113 Ljubljana, Slovenija kovacic.andreja@gmail.com	Matevž Ogrinc Fakulteta za računalništvo in informatiko Večna pot 113 Ljubljana, Slovenija mo0429@student.uni-lj.si	Anja Hrovatič Fakulteta za računalništvo in informatiko Večna pot 113 Ljubljana, Slovenija ah7651@student.uni-lj.si
---	--	--

Povzetek

Nove generacije igralnih konzol postajajo vse bolj zmogljive, podpirajo vse več storitev in zaradi tega hranijo vse več podatkov, ki jih je možno koristiti v forenzični preiskavi. Prav tako so postale tudi omrežne naprave z možnostmi brskanja po spletu, pošiljanja elektronskih sporočil in s tem postale bolj privlačne tudi za zlonamerno uporabo. Članek oriše postopke preiskave Nintendo Wii konzole, kjer preiskovalec lahko najde obilo informacij o zlonamerni uporabi. Tekom članka avtor opiše same značilnosti Nintendo Wii konzole in aplikacije, ki lahko preiskovalcem prispevajo k forenzični preiskavi ter se ob tem dotakne težav z ekstrakcijo le-teh. V pričujočem delu smo predstavili področje preiskave igralnih konzol, razširjeno povzeli značilnosti analize in zanimive kanale Nintendo Wii konzole ter njihove praktične implementacije.

Ključne besede

forenzična analiza, digitalni dokazi, igralna konzola, Nintendo Wii

1. UVOD

Igralne konzole, kot so Microsoft Xbox 360, Sony Playstation 3 in Nintendo Wii so postale omrežne medijske platforme in s tem nadomestile marsikatero uporabnikovo potrebo po tradicionalnem računalniškem sistemu. Omogočajo vedno večjo povezljivost s funkcionalnostmi kot so uporaba spletnega brskalnika, e-mail sporočanje, instant sporočanje in VoIP. S tem pa imajo tudi vedno večji potencial za zlorabo. Nintendo Wii je med igralnimi konzolami sicer izmed manj privlačnimi tarčami za zlonamerno uporabo, predvsem zaradi slabše zmogljivosti in majhne kapacitete diska.

V naslednjih razdelkih bomo opisali razumevanje Nintendo Wii-ja s forenzične perspektive, opisali zmožnosti naprave ter metode za forenzično preiskovanje, kakšne podatke lahko pridobimo iz konzole, kakšni so problemi z ekstrakcijo po-

datkov ter potencial Nintendo Wii-ja v forenzični analizi, njegove omejitve in potencial za zlorabo.

2. PREDSTAVITEV PODROČJA

Igralne konzole so že od leta 1972 in do danes eden izmed glavnih načinov preživljjanja prostega časa ter so dostopne širši množici. Zaradi tesne povezave z računalništvom in internetom je vedno več možnosti zlorabe konzol. Hkrati pa prinašajo čedadje več informacij in dokazov, ki lahko pripomorejo k forenzični preiskavi.

V članku [14] so izvedli forenzično analizo konzole Nintendo Wii U. Prikazali so kako se iz konzole pridobjijo relevantni podatki kot so spomin in datoteke. Iz spomina so pridobili datotečne sistemski poti in zgodovino brskalnika. Analizirali so posamezne funkcionalnosti Nintenda in njihov potencial v forenzični preiskavi. Na podlagi pridobljenih podatkov so zasnovali postopek za forenzično analizo. Njihova rešitev je prosto dostopna in odprtokodna. V [12] so razvili forenzično metodologijo analize Nintendo 3DS, katera forenzičnim preiskovalcem pomaga maksimizirati ekstrakcijo dokazov in minimizirati, preprečiti, uničenje podatkov.

Za forenzično analizo so zanimive tudi druge igralne konzole kot sta PlayStation in Xbox. Najnovejša različica igralne konzole PlayStation omogoča brskanje po spletu, prenašanje datotek in klepet. Naštete funkcionalnosti so zanimive za forenzične preiskovalce in so potencialen vir informacij. V [8] identificirajo vire informacij s forenzičnim pomenom in predlagajo metode za pridobitev podatkov na zanesljiv način. V okviru [9] so raziskovali fizične modifikacije konzole Xbox 360, tehnikе za pridobitev slike diska in na podlagi tega razvili smernice za forenzično preiskavo.

3. VSEBINA

V pričujočih razdelkih bomo povzeli pristop k forenzični preiskavi konzole Nintendo Wii.

3.1 Koristne značilnosti konzole Nintento Wii v forenzični preiskavi

Nintendo Wii (1) je izmed igralnih konzol med manj privlačnimi tarčami za zlorabo. Razlogi za to ležijo v njegovi slabši zmogljivosti ter kapaciteti trdrega diska. Poleg tega Nintendo Wii vsebuje lastniški datotečni sistem, ki je zaprt za preiskavo tudi ob obnovitvi. Prav tako preiskovalci ne morejo zagotoviti forenzične integritete z uporabo zaščite zapisova-

nja (ang. *write protection*). Naštete lasnosti predstavljo manjšo fleksibilnost pri forenzični analizi.



Figure 1: Konzola Nintendo Wii.

Sestavna dela konzole sta Samsung čip, ki nudi 256MB flash spomina [1] in SD čitalec kartice, ki omogoča uporabo dodatne SD kartice in se lahko uporabi za zunano analizo. Glavni spomin je pritrjen na matično ploščo in se ga ne da preprosto odstraniti. Nintendo Wii je torej relativno zaprt lastniški sistem, ki je hkrati koristen in omejujoč s pogleda forenzične preiskave. Koristen v smislu, da preiskovalcem že na začetku zmanjša število različnih tipov programske opreme, ki jo lahko uporabijo pri preiskavi, hkrati pa tudi zmanjšuje število različnih oblik zlorab sistema. Zaprtost sistema zagotavlja tudi nezmožnost brisanja avtomatsko beleženih podatkov - dokazov. Forenzična preiskava je omejena tudi na napravah, ki se lahko analizirajo ločeno, npr. dodatna SD spominska kartica.

Kljub zgoraj naštetim omejitvam, še zmeraj obstaja potencial za zlorabo konzole, in sicer to omogočajo naslednje funkcionalnosti: zmožnost brezžične povezave, sporočanje podobno e-mailu, spletno nakupovanje, avtomatsko beleženje uporabe naprave. V forenzični preiskavi se preiskovalci torej osredotočajo na analizo teh funkcionalnosti na naslednje načine:

- Brezžična povezava
Nintendo Wii je omrežna naprava, zato se lahko za eliminacijo ali potrditev v mehanizmih za beleženje uporabi MAC naslov naprave.
- Sporočanje podobno e-mailu
Funkcionalnost omogoča komunikacijo med različnimi Wii napravami ter omogoča tudi prejemanje in pošiljanje na e-mail naslove s predhodno avtorizacijo. Preiskovalci lahko z analizo komunikacije najdejo povezave med posameznimi uporabniki oziroma napravami.
- Brskanje s spletним brskalnikom
Nintendo omogoča brskanje po spletu z brskalnikom, kar forenzičnim preiskovalcem omogoča analizo uporabe interneta. To je eden izmed glavnih razlogov za analizo Nintedo Wii-ja. Takšna analiza ni vedno mogoča, saj je funkcionalnost plačljiva in se zato ne pojavi na vseh sistemih.

- Spletno nakupovanje

Konzola omogoča spletno nakupovanje z uporabo kreditne kartice in s tem beleženje podatkov o spletnih nakupih.

- Avtomatsko beleženje

Gre za avtomatsko beleženje dnevne uporabe konzole. Ker podatkov uporabnik ne more izbrisati, bi se le-ti lahko uporabili za potrditev ali ovržbo izjav posameznikov o dogodkih ali drugih dokazov.

3.2 Analiza konzole Nintendo Wii

V zgornjih razdelkih smo že omenili, da je ena izmed večjih pomanjkljivost Nintenda majhen spomin, kar preiskovalce omejuje pri analizi, saj ne morejo uporabiti klasičnih forenzičnih orodij za analizo kot sta EnCase ali FTK. Potencialen problem pri analizi je tudi kompleksnost interpretacije, saj ni nujno, da je Nintendo Wii primarni vir dokazov.

Za ekstrakcijo podatkov znotraj sistema bi tako lahko uporabili Wii zagonski disk, ki je zmožen dostopati do in skopirati podatke direktno na spominsko kartico. Takšen disk sicer ne obstaja v javni domeni in bi ga morali, zaradi lastniškega sistema, razviti v sodelovanju z Nintendo Inc. Drug možen način za forenzično analizo pa je fizična odstranitev notranjega spomina in izvedba ekstrakcije podatkov na odstranjenem disku. Takšen način je v forenzičnih preiskavah bolj tradicionalen, saj pridobimo podatke v njihovem najbolj surovem formatu in ti niso podvrženi interpretaciji. Ekstrakcija podatkov s to metodo terja višjo raven znanja, spremnosti in virov.

V članku [13] so analizo konzole Nintendo Wii izvedli kot analizo kompleksne vgrajene naprave (ang. *embedded device*). Takšen pristop namreč ponuja najbolj primerno forenzično metodologijo za preiskavo. Cilj preiskave je posneti vse aktivnosti in s tem beležiti vse morebitne spremembe sistema in jih, če je le možno, zminimizirati. Preiskovalec je ob analizi beležil interakcije z vsemi kanali (ang. *channels*), ki bi lahko povzročile spremembo sistema. Priporočeno je, da se analiza konzole izvaja vsaj en dan po uradnem zajemu, da obstoječe podatke izoliramo, saj *Wii Play Time* funkcija avtomatsko beleži dnevno uporabo sistema. Kot alternativo za izolacijo podatkov, lahko preiskovalec spremeni nastavitev časa v napravi na čas v prihodnosti. Slednji pristop je manj zaželen, saj lahko pride do sprememb podatkov potencialnih za uporabo.

V okviru članka so razvili strukturo za forenzično analizo in jo definirali na naslednji način:

- Aktivacija zunanjega mehanizma za beleženje ali naprave za snemanje,
- odstranitev SD kartice - neodvisna preiskava,
- prižig Wii konzole,
- preiskava diska,
- preverjanje nastavitev, določitev trenutnega časa in pridobitev informacij o povezanih brezžičnih omrežjih

- preverjanje sporočil na oglasni deski (ang. *board*), preiskava sistema za sporočanje, določitev uporabe sistema za pomembne datume, pregled poslanih sporočil, preverjanje imenika,
- preiskava kanala Mii in drugih kanalov,
- ponovno preverjanje deske s sporočili in beleženje sprememb.

Struktura preiskave zagotavlja, da preiskovalec zabeleži vsa kršne spremembe povzročene na sistemu in jasno opredeli originalne ter končne rezultate. S tem se omogoči tudi nadaljnjo oziroma ponovno analizo konzole, če je ta potrebna.

3.3 Podrobnejši pregled faz

V pričujočem razdelku bomo povzeli opise posazmeznih faz, kot so jih podali raziskovalci, od prižiga konzole naprej.

3.3.1 Prižig Wii konzole - Aktivacija sistema

Ob zagonu naprave potrebujemo kontroler, da lahko izberemo tipko A. Ob tem vstopino v glavni meni (2), preko katerega raziskovalec dobi vpogled v uporabo sistema. Če dostopamo do katerekoli ikone, s tem potencialno sprememimo *Wii Play Time* funkcijo.



Figure 2: Glavni meni konzole Nintendo Wii.

3.3.2 Disk

Prva ikona v prvi vrsti predstavlja trenutni disk v napravi. Ta je prazen disk ali disk trenutne igre, če naprava ni bila spremenjena. Preiskava tega elementa prinese malo spoznanj. Če želimo dostopati do podatkov o disku, ni potrebno direktno dostopati do diska. Informacije so na voljo v podmeniju, ki ne bo spremenil *Wii Play Time* sporočila.

3.3.3 Preverjanje nastavitev

Za pregled nastavitev izberemo Wii logo v spodnjem levem kotu. Od tam imamo na voljo 2 možnosti: nastavitev sistema (ang. *System Settings*) in podatkovne možnosti (ang. *Data Management*). V nastavitevah sistema lahko razberemo različico operacijskega sistema, čas in datum, nastavitev povezave, podatke o konzoli, *agreement/contact* in internet. Pri pregledu časa mora biti raziskovalec pozoren na to, da konzola nima samodejnega prilagajanja na poletni/zimski čas in da je možno čas poljubno spremnijati. Za forenzično preiskavo je pomembnejša možnost internet, kjer so na voljo 3 možne omrežne nastavitev. Iz tega lahko razberemo na

kakšna omrežja se je konzola povezala in podrobnosti vsake od teh. Iz tega je možno napravo umestiti v fizično okolje.

Avtorji članka podajo podrobnejša navodila o preiskavi brezžičnih povezav. Če izberemo to, so nam na voljo 4 možnosti: uporabi to povezavo, test povezave, izbira nastavitev, počisti nastavitev. Iz vidika forenzične preiskave je neugodno počistiti nastavitev. Možnost spremeni nastavitev nam poda trenutne nastavitev povezave. Skozi nastavitev se pomikamo s pritiskanjem na puščice ob strani in se pri tem sprehodimo čez SSID omrežja, tip povezave, podatke o enkripciji in podatke, ki jih potrebujemo za povezovanje. Za forenzično preiskave je zanimiv pregled tipa enkripcije, kjer se sicer prikaže maskirano geslo, vendar iz tega lahko ugotovimo njegovo dolžino. Po tem si lahko preiskovalec ogleda nastavitev IP naslova, razen če se ta nastavlja samodejno. Zanimiva je še možnost podatki o konzoli, kjer najdemo MAC naslov naprave (napisan sicer tudi na na napravi sami) in MAC naslov LAN adapterja, če je ta povezan.

Meni upravljanje s podatki preiskovalcu pove, kateri podatki so shranjeni na napravi. Prikaže podatke na Wii spominu in morebitni SD kartici. SD kartico je potrebno pregledati tako iz Wii naprave kot tudi kot samostojen vir. Meni shrani podatke (ang. *Save Data*) prikaže vse podatke na napravi ali pa vse priklopljene Game Cube igre (ang. *cartridges*). Vse prenešene aplikacije kot so igre, internetni kanal in podobno so označene kot kanali.

3.3.4 Preiskovanje sporočil

Nintendo je ustvaril svoj sporočilni sistem *Wii Message Board*. Vsaka konzola ima svojo unikatno številko, ki služi za identifikacijo naprav. Wii lahko uporabimo za pošiljanje sporočil med Wii napravami in elektronsko pošto. Preiskavo pričemo na ikoni *Wii Message Board*, kjer vidimo seznam nedavnih sporočil in e-mailov. Za lažji pregled si lahko pomagamo s koledarjem. Možna je tudi preiskava imenika uporabnikov, s katerimi si je uporabnik konzole izmenjal sporočila. To storimo skozi kreiranje novega sporočila, kjer se na desni strani izpišejo tisti, ki jim je uporabnik predhodno že pisal. Poleg sporočil samih, Nintendo shranjuje tudi metapodatke o sporočilu in sistemu. Najbolj pogosto so to podatki o uporabi v posameznem dnevu (ang. *Today's Play History*). Ta sporočila sicer ne hranijo uporabnika ali kdaj se je igranje začelo/končalo, vendar jih za razliko od običajnih sporočil, ni mogoče izbrisati.

3.3.5 Kanal Mii

Mii so prilagodljivi uporabnikovi liki, ponavadi narejeni po videzu uporabnika, katere lahko uporabi pri interakciji z določenimi igrami. Do njih dostopamo s klikom na drugo ikono v vrhnji vrstici in nato s klikom na start. Glavni meni prikaže vse Mii like. Za podrobnejši pogled Mii lika je potreben klik na piščalko in nato klik na posamezen Mii lik.

Kljub temu da sami po sebi ne prispevajo pri forenzični preiskavi, lahko preiskovalcu pomagajo odkriti število uporabnikov sistema. Pri primeru [6] v Veliki britaniji je zaradi dodatnega Mii lika prišlo do ugotovitve ženine nezvestobe. Poleg tega so Mii liki dostopni preko prej omenjene naslovne knjige, kar lahko pripomore k pridobitvi več informacij o liku. Vendar pa Mii kanal vpliva na sporočila o času igranja, kar pomeni, da je potrebno pred kakršnokoli interakcijo

shraniti in zapisati stanje naprave.

3.4 Dodatni kanali

Poleg zgoraj omenjenih kanalov lahko nakupi preko kanala Wii trgovine vplivajo na izgled glavnega menija. Že obstoj teh kanalov lahko prispeva k večjemu vpogledu v uporabo sistema. Kanali kot so kanal vremenske napovedi (ang. *Forecast Channel*), kanal novic (ang. *News Channel*) in internetni kanal (ang. *Internet Channel*) preiskovalcu povejo, da je vsaj enkrat prišlo do povezave z internetom. Kljub temu da vsi našteti kanali ne pripadajo osnovnemu paketu, vsak izmed njih vpliva na sporočila o času igranja.

3.4.1 Kanal Internet

Iz forenzičnega vidika je spletni brskalnik Wii odličen vir informacij in hkrati velika ranljivost. Brskalnik, sam po sebi, ima pomanjkljivo implementacijo "Internet Favourites" in nobenega načina beleženja zgodovine, kar oteži pridobivanje pomembnih forenzičnih informacij. Obe pomanjkljivosti lahko povežemo s primanjkljajem prostora na trdem disku.

Kljub temu lahko na drugačne načine pridobimo informacije, namreč "Internet Favourites" shranjuje sličice obiskanih naslovov, katere lahko s pomočjo zunanje opreme in ugibanjem uporabimo pri ugotovitvi spletnega naslova. Na žalost je tak pristop preveč nenatančen in je zaradi večje zanesljivosti uporabljen bolj tehnična metoda, kjer preiskovalec poveže Nintendo Wii na internetno omrežje ter klikne na povezavo do zabeleženega priljubljenega spletnega mesta. Z uporabo Kismet [10] za prislушкиvanje paketov in Wireshark [11] za analizo paketov, lahko preiskovalec identificira Nintendo napravo 'Nintendo_XX:XX:XX', kjer X-i ponazarjajo MAC naslov naprave.

3.4.2 Kanal za napoved vremena

Iz forenzičnega vidika kanal za napovedovanje vremena v veliki večini služi le kot izboljšan način analize relacije med geografsko lokacijo naprave in uporabnika.

3.4.3 Kanal Wii trgovine

Spletna trgovina je na splošno dober vir informacij za preiskovalca, vendar v primeru Kanala Wii trgovine, ki ne shranjuje uporabnikovih podatkov in podatkov o zgodovini nakupov, je le-ta za forenzične preiskave nepomembna. Moramo pa omeniti, da to ne izključuje možnosti, da Nintendo Co Ltd. ne shranjuje podatkov o svojih uporabnikih.

3.4.4 Kanal novic in kanal slik

Oba kanala forenzičnim preiskavam ne pripomoreta veliko dokazov. Kanal novic je namenjen tekstovnemu ogledu aktualnih novic, razdeljenih na več kategorij, kanal slik pa je aplikacija namenjena ogledovanju slikovnih datotek. Primarno kanal slik ne prispeva preiskavi, vendar njegov obstoj poveča možnost obstoja SD spominskih kartic.

Hkrati moramo omeniti še ponastavljanje na osnovne nastavitev (ang. *Restore Factory Settings*), ki v primeru zagona med analizo izbriše večino dokazov na napravi.

3.5 Zunanje modifikacije

V osnovi je integracija zunanjih modifikacij ali modifikacijskih čipov (ang *mod-chipov*) proti originalnemu namenu uporabe konzole. Namreč v primerjavi s stacionarnimi računalniki, konzole ne morejo pogosti arbitrarne kode. Inštalacija le-teh v večih primerih namigujejo na ilegalne aktivnosti kot so kopiranje in igranje iger brez avtorizacije. Kljub temu je pomembno za preiskovalce znanje o teh nadgradnjah in kako vplivajo na podatke in dokaze pridobljene z originalnih naprav. Obstaja več različnih nadgradenj, ki v osnovi delujejo na isti način.

Na primeru spletne strani Wii-ModChips.com so mod-čipi fizično dodani na DVD vhod in prestrezajo kljice glavne enote. Uporaba le-teh preskoči validacijo diska in v novejših primerih podpirajo poganjanje neznane kode [7]. Hkrati lahko uporaba takih nadgradenj pomaga tudi forenzični preiskavi. Projekt *WiiLi* skuša integrirati Linux sisteme v Wii. Do sedaj še ni bilo najdenega načina integracije, vendar če projektu uspe, lahko odpre mnogo več načinov pridobivanja informacij in dokazov pri preiskovanju Wii konzole.

Trenutno še ni enostavnega načina dokazovanja ali je v konzoli prisotna nadgradnja, brez da konzolo fizično odpremo z uporabo tri-stranskega izvijača. Poleg tega lahko ugibamo o uporabi nadgradenj glede na prisotnost ilegalnih kopij Nintendo Wii iger v okolini naprave. Preiskovalci so z uporabo popularne nadgradnje *WiiKey* [5] žeeli ugotoviti način uporabe in vpliva nadgradnje na prej omenjene podatke. Potrdili so, da je kljub uporabi mod-čipa naprava zaganjala originalne igre. V primeru ilegalnih iger je bil potreben nov zagonski disk. Ob zagonu omenjenega diska je Nintendo Wii ugotovil, da je igra od konzole *Game cube* in posledično je bil zagnan Wii z drugačnim glavnim menijem. Z novim glavnim menijem so ilegalne igre delovale kot originalne. Hkrati so se ohranile vse prej omenjene funkcije, kar preiskovalcu ne oteži dela.

Posledično uporaba nadgradenj, v določenih primerih, ne vpliva pretirano na sistemski nastavitev in hranjenje podatkov. Vendar lahko v primeru, kot je integracija linux sistema, take nadgradnje spremenijo celotno delovanje Nintendo Wii in odprejo več možnosti ilegalne uporabe, kar pa bi potrebovalo prilagojeno metodo analize.

4. PRAKTIČNE IMPLEMENTACIJE

Na centru za kibernetično forenziko organov kazenskega preona (ang. *Law enforcement Cyber center*) se zavedajo groženj in morebitnih zlorab, ki jih prinašajo igralne konzole. V ta namen so NW3C, *the National White Collar Crime Center*, razvili spletni vir, ki vsebuje informacije o igralnih konzolah in ročnih igralnih napravah, ki lahko vsebujejo digitalne dokaze. V njem so zbrane vse tehnične specifikacije, zmožnosti in rezultati forenzičnih preiskav različnih igralnih konzol. Vir je prostost dostopen registriranim uporabnikom [2].

V Združenem Kraljestvu so se že v letu 2009 zavedali prilžnosti, ki jih prinaša Nintendo Wii. Začeli so izvajati izobraževanje policistov, kako izslediti zločince s pomočjo konzole. Dva osumljence lahko namreč trdita, da se ne poznata, obstaja pa med njima povezava na Nintendu Wii, ki dokazuje, da sta znanca. Na NPIA, *the National Policing Improvement Agency* se tako zavedajo, da igralne konzole vsebujejo

potencialne dokaze [3]. Tudi v Ameriki so začeli uporabljati igralne konzole v svojih preiskavah zločinov. Fokusirajo se predvsem na Xbox in Playstation 3 ter se zavedajo pomembnosti podatkov zabeleženega igranja (ang. *game log*), časovnih žigov in drugih, ki lahko potrdijo ali ovržejo alibi osumljencev. V svojih preiskavah sodelujejo z Microsoftom, ki jim zagotavlja dostop do dodatnih podatkov o osumljencih [4].

5. ZAKLJUČEK

Pričujoče delo je namenjeno vpogledu in možni uporabi Nintendo Wii in drugih konzolnih sistemov kot vir dokazov za digitalno forenzično preiskavo ilegalne dejavnosti.

Z nadaljnimi raziskavami lahko ugotovimo več možnih načinov preiskovanja konzolnih sistemov, vendar bodo zaradi prostorske omejenosti le-teh v večini vključevale interakcijo z napravo, saj konzolni sistemi vsebujejo svoj poseben in zaprt operacijski sistem, katerega je potrebno uporabljati za pridobivanje forenzičnih podatkov. Dve najpomebejši funkciji sta komunikacijska mehanizma, ki ju ponuja Nintendo Wii sistem. Mehanizma se z internetom konstantno nadgrajuje in se lahko tekom časa razširita tudi na druge sisteme. Hkrati se podatki o e-naslovih shranjujejo na strežnikih od podjetja Nintendo, kar spodbuja kooperacijo preiskovalca s podjetjem.

Poleg tega je potrebno omeniti, da nespremenljivost fizične naprave še ne pomeni nespremenljivost programske opreme. Z internetom se namreč programska oprema konstantno nadgrajuje ter spreminja. Hkrati delo opisuje le eno zunanjmo modifikacijo, ki se s časom nadgrajuje, spreminja in ima vendno večji vpliv na operacijski sistem konzolne naprave. V realnosti obstaja vedno več nadgradenj, ki lahko razširijo možnosti ilegalne uporabe konzolnih naprav in tako odprejo tudi več načinov uporabe konzolnih sistemov kot vir digitalnih dokazov za forenzično preiskavo.

Delo tudi zanemari glavni dodatek Nintendo Wii konsole - brezzični daljinski upravljalnik, ki vsebuje vgrajen spomin in bi lahko v nadaljnjih raziskavah pripomogel kot dodatni vir dokazov za preiskavo. Iz dela vidimo, da fizične igralne konzole niso več le enostavne naprave, katerih primarni namen je le igranje iger, vendar se postopoma širijo tudi na ostala področja računalništva in bodo vedno več prispevale kot vir dokazov za digitalno forenziko in analizo.

6. REFERENCES

- [1] Cracking open the nintendo wii.
<https://www.techrepublic.com/pictures/cracking-open-the-nintendo-wii/>. [Dostopano: 29. 4. 2019].
- [2] Gaming consoles.
<http://www.iacpcybercenter.org/officers/cyber-forenics/>. [Dostopano: 29. 4. 2019].
- [3] Police trained to use nintendo wii to catch criminals.
<https://www.telegraph.co.uk/technology/video-games/nintendo/6252705/Police-trained-to-use-Nintendo-Wii-to-catch-criminals.html>. [Dostopano: 29. 4. 2019].
- [4] Us police use games consoles in crime investigations.
<https://nakedsecurity.sophos.com/2012/01/26/us-police-use-games-consoles-in-crime-investigations/>. [Dostopano: 29. 4. 2019].
- [5] Wikey. <https://wiki.gbatemp.net/wiki/WiKey>.

[Dostopano: 28. 4. 2019].

- [6] . b. r. . C. . 22 November. Wii have caught you cheating on your husband.
- [7] Bushing. *Console Hacking: State of the Wii*. 2007.
- [8] M. Davies, H. Read, K. Xynos, and I. Sutherland. Forensic analysis of a sony playstation 4: A first look. *Digital Investigation*, 12:S81–S89, 2015.
- [9] H. A. B. M. ISA. The practical analysis towards developing a guideline for the xbox 360 forensic. 2009.
- [10] M. Kershaw. Kismet wireless. *Retrieved from the Web*, 2007.
- [11] A. Orebough, G. Ramirez, and J. Beale. *Wireshark & Ethereal network protocol analyzer toolkit*. Elsevier, 2006.
- [12] H. Read, E. Thomas, I. Sutherland, K. Xynos, and M. Burgess. A forensic methodology for analyzing nintendo 3ds devices. In *IFIP International Conference on Digital Forensics*, pages 127–143. Springer, 2016.
- [13] B. Turnbull. *Forensic investigation of the Nintendo Wii: A first glance*. PhD thesis, Purdue University Cyber Forensic Lab, 2008.
- [14] R. van Dijk and D. Geist. Forensic analysis of the nintendo wii u. 2016.



4 PROTIFORENZIKA IN POKVARJENO GRADIVO ANTIFORENSICS AND DAMAGED MATERIAL

Ralph Palutke, Felix Freiling: Styx: Counteracting Robust Memory Acquisition (DFRWS EU 2018)*

Povzetek članka in pregled področja

Tilen Tomakić
Fakulteta za računalništvo in
informatiko
tt5157@student.uni-lj.si

Nenad Babić Bodiroža
Fakulteta za računalništvo in
informatiko
nb9862@student.uni-lj.si

Danijel Maraž
Fakulteta za računalništvo in
informatiko
dm9929@student.uni-lj.si

POVZETEK

Članek na kratko predstavi splošno področje pridobivanja spominskih slik digitalnih naprav z uporabo forenzičnih orodij, ter vsebino članka Ralph Palutke, Felix Freiling: Styx: Counteracting Robust Memory Acquisition (DFRWS EU 2018) [3].

Tekom procesa pridobivanja dokazov v postopku digitalne preiskave, preiskovalci pogosto uporabljajo raznovrstne forenzične tehnike ter forenzična programska orodja. Vse to z namenom pridobivanje natančne slike glavnega pomnilnika obravnavanih digitalnih naprav. Pridobljene slike nato uporabljajo za ekstrakcijo digitalnih dokazov in identificiranje naprednih groženj. Med najbolj razširjeno in uporabljenou tehniko naprednega pridobivanja pomnilniških slik spada orodje **pmem**, ki sta ga razvila Johannes Stütten in Michael Cohen (Google). Izbran članek predstavlja t.i. proof-of-concept sistem imenovan Styx. Omenjen sistem z zakrivanjem sledi, preiskovalcem izniči učinkovitost uporabe predhodno omenjenega orodja **pmem** in preostalih klasičnih orodij, namenjenih pridobivanju pomnilniških slik. Implementacija sistema Styx sestoji iz naložljivega jedrnega modula, ki omogoča spodbopavanje 64-bitnih Linux sistemov z uporabo Intelove VT-x virtualizacije ter njene razširivite EPT.

Splošni izrazi

Robust Memory Acquisition

Ključne besede

Robust Memory Acquisition Incriminating Evidence Virtualization Linux Intel Styx

1. UVOD

Izbran članek govori o načinu skrivanja zlonamerne kode na računalniku. Zaradi vse pogostejšega skrivanja in poganganja zlonamerne programske kode v pomnilniku računal-

*Povzetek članka in pregled področja.
Seminarska naloga pri predmetu Digitalna Forenzika.

nika je med digitalnimi preiskavami vse bolj razširjena uporaba različnih digitalnih forenzičnih orodij, ki poskrbijo za pridobivanje pomnilniških slik oz. podatkov (dokazov) iz pomnilnika. Omenjena orodja po pridobitvi pomnilniških slik podatke zapisujejo na zunanje priklopne podatkovne enote, kot so med drugim USB ključi (trenutno najbolj razširjeno orodje za pridobivanje podatkov iz pomnilnika je orodje "pmem"). V članku je opisana zasnova programske opreme z imenom Styx, ki skrbi za skrivanje in zagotavljanje zasebnosti podatkov, ki se hranijo v pomnilniku, pred predhodno omenjenimi orodji za pridobivanje podatkov ter protivirusnimi orodji. Članek razišče možnost skrivanja in shranjevanja programske kode v skriti del pomnilnika. Ker skriti del pomnilnika ni namenjen neposredni uporabi s strani operacijskega sistema, veliko orodij pri analizi pomnilnika ne upošteva skritih regij. Ko govorimo o skritem pomnilniku govorimo o rezerviranem delu pomnilnika namenjenega PCI napravam. Omrežna kartica in druge naprave se lahko na primer poslužujejo pomnilnika za začasno shranjevanje TCP paketov. Rezerviran pomnilnik določi BIOS ob vklopu računalnika. Predstavljen program Styx je v resnici naložljiv Linux jedrni modul, ki prestavi gostujuči operacijski sistem na virtualiziran način [1], sam Styx pa deluje kot lahek hipervisor. S tem lahko svoje delovanje in podatke učinkovito skrije.

2. SLOVAR KRATIC

- EPT - Extended Page Table
- USB - Universal Serial Bus
- PCI - Peripheral Component Interconnect (device)
- VMM - Virtual Machine Monitor
- TCP - Transmission Control Protocol
- BIOS - Basic Input/Output System
- RAM - Random Access Memory
- ROM - Read Only Memory
- LKM - Linux Kernel Module
- VMX - Virtual Machine Extensions
- MMU - Memory Management Unit
- DMA - Direct Memory Access

3. PREDSTAVITEV PODROČJA

Prvi računalniški virus je bil napisan leta 1982 s strani 18 letnega Rich Skrenta iz Pittsburgha in je svojo kodo imel skrito v zagonskem sektorju računalnika. Bil je prvi opažen resni pojav zlonamerne programske kode (ang. malware) in se je širil s floppy disketami. Praktično večino virusov vse do zdognjih 2000 je imelo svojo kodo na trdem disku, saj je velikost in dinamična narava RAM-a preprečevala zasidranje škodljivih programov. Ti so se morali precej razviti v zapletenosti, da so bili zmožni imeti le manjšo prisotnost na samem trdem disku in so svojo kodo tako rekoč vsakič ob zagonu oziroma podobnemu dogodku za boljše delovanje ustvarili v RAM-u. Velik razlog za migracijo na RAM je bila tudi vsespolna razširjenost protivirusnih programov, ki so se primarno specializirali na iskanje virusov na trdem disku. Ob koncu prvega desetletja se je računalništvo začelo bistveno spremenjati. Vse bolj je postajal pomemben koncept virtualizacije ob enem skupaj z rastjo oblačnih storitev. Bilo je namreč kar nekaj primerov virusov, ki so s pomočjo orodji kot je VirtualBox bivali v svetu virtualizacije. Torej neposredno na virtualiziranem podsistemu ali celo v sami programski opremi virtualizacije. Za računalniške sisteme to še danes predstavlja resno grožnjo, saj je programska oprema, ki omogoča virtualizacijo precej prepredena s ključnimi funkcijami računalnika in zahteva visoko stopnjo privilegiranosti za izvajanje. Posledično se zlonameri programi lahko izognejo večini primitivnejšim protivirusnim programom.

Za razliko od teh so sodobni protivirusni programi bolj pametni in preiščejo večino pomnilniškega prostora, ter uporabljajo zapletene hevristike za zaznavanje zlonamerne kode. Posledično ni več dovolj, da se zlonamerne koda le naloži na pomnilnik, ampak je bistvenega pomena tudi tako imenovana zamglitev (ang. obfuscation) zlonamerne kode [2]. Torej, da je v osnovi težko razbrati kako deluje program oziroma ali sploh program obstaja. Največjo iluzijo nam pa zagotovo predstavlja prav tehnologija popolne virtualizacije. Celoten operacijski sistem se da preklopiti na tankega supervizorja (ang. VMM), ki se zaveda vseh pisalno bralnih operacij sistema in se lahko nanje ustrezeno odzove, da zaščiti svojo zlonamerne kodo. Izkazalo se je, da je tudi sam preklop operacijskega sistema relativno preprost, saj vsi sodobni proizvajalci računalniške opreme omogočajo neko stopnjo virtualizacije [1].

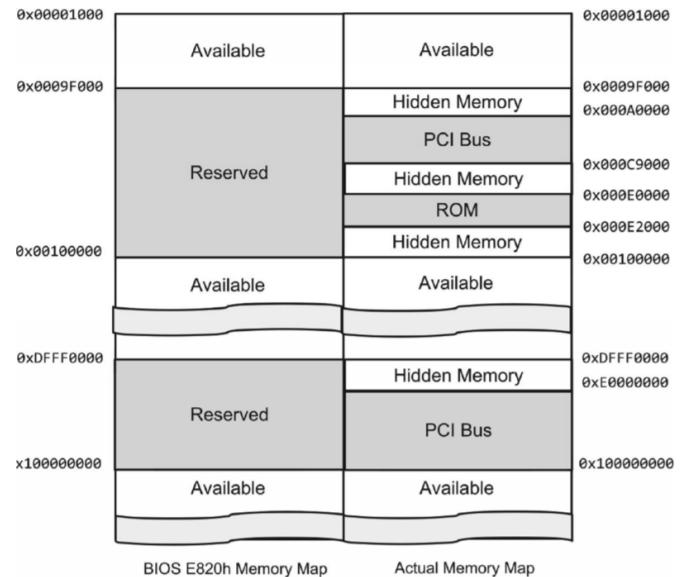
4. TEHNIČNO OZDAJE

Arhitektura sistema Styx sestoji iz več pod komponent, pri njegovem nalaganju pa moramo biti pozorni predvsem na tri ključne komponente opisane spodaj.

4.1 Naložljivi jedrni moduli

Vstavljanje naložljivih jedrnih modulov je priročna funkcionalnost. Jedru omogoča majhnost ter fleksibilnost in hkrati podpira vrsto različne strojne opreme. Ker se jedrni moduli izvajajo v privilegiranem načinu se pogosto uporablja preverjanje podpisa modula. Zato Styx uporablja tehniko opisano v članku *Robust Linux Memory Acquisition with Minimal Target Impact* [5] za nalaganje nepodpisanih modulov. Na ta način Styx prelisiči jedro v izvedbo svojega modula s polnimi pravicami.

To stori z metodo vsiljevanja škodljivega modula v že obstoječe naložen modul v gostujučem jedru. Rezultat je polno



Slika 1: Primer skritega pomnilnika na 4GB pomnilniku [3]

zdržljiv modul v že pognanem jedru.

4.2 Rezerviran pomnilniški prostor

Fizični pomnilniški naslovni prostor obsega vse naslove, ki se pojavi na pomnilniškem vodilu (Figure 1). Vendar fizični naslovni prostor vsebuje več kot le RAM. Ko se računalnik zaganja, BIOS ustvari zemljevid uporabnega in rezerviranega pomnilnika. Rezervirani pomnilnik je namenjen ROM-u, PCI napravam in samemu BIOS-u. [4]. Ker običajna orodja za analizo pomnilnika ne upoštevajo rezerviranega pomnilnika, je ta priljubljeno skrivališče za zlonamerne programe.

4.3 Virtualizacija OS med delovanjem

Virtualizacija je tehnika s katero računalniške vire abstrahiramo. Z virtualizacijo virov, lahko poganjam več operacijskih sistemov hkrati nad enakimi viri [1]. Hipervizor je program, ki skrbi za abstrakcijo naprav in emulacijo le teh. Hipervizorje delimo na dve kategoriji:

- Tip 1 - Hipervizorji, ki tečejo na goli strojni opremi (ang. bare hardware).
- Tip 2 - Hipervizorji, ki tečejo nad gostiteljevim operacijskim sistemom.

Program Styx vsebuje hipervizor tipa 1, saj se postavi pod gostujoči operacijski sistem in s tem pridobi popoln nadzor.

5. STYX

Zanimiva lastnost sistema Styx je, da računalnika pri nalaganju in skrivanju Styxa ni potrebno niti ponovno zagnati, ker se Styx računalniku predstavi kot naložljiv Linux jedrni modul (LKM). Skrivanje navadno delimo na dva koncepta oz. metodologiji. Prva temelji na skritem (rezerviranem)

pomnilniku. Druga pa uporablja Intelovo tehnologijo virtualizacije VMX.

Styx najprej analizira pomnilnik in določi potencialno prost rezerviran pomnilnik PCI naprav. Na neuporabljen del rezerviranega pomnilnika naloži svojo kodo, prične s postopkom prestavitev operacijskega sistema na virtualiziran način (Intel VMX) in po uspešni naložitvi požene nad operacijskim sistemom program Eraser LKM, ki odstrani Styx jedrni modul.

Glavni cilj virtualizacije gostujučega operacijskega sistema je nadzor branja pomnilnika kjer se nahaja Styx. To je mogoče, ker je gostujuč operacijski sistem v celoti virtualiziran (ne teče več na ring 0), s tem pa lahko Styx uporabi Intel EPT tehnologijo za preslikavo fizičnih naslovov na virtuelne. Posledično lahko učinkovito nadzira dostop do branja izbranih regij pomnilnika.

5.1 Arhitektura

Arhitektura Styx rootkita, je sestavljena iz 3 komponent:

- **Nameščevalne komponente**, ki teče v uporabniškem načinu. Nadzira jedrne komponente preko 'ioctl' linux vmesnika.
- **Rootkit LKM**, ki je zadoložen za namestitev hipervizorja in virtualizacijo ciljnega sistema.
- **Eraser LKM**, ki izbriše vse sledi iz pomnilnika sedaj že gostujučega operacijskega sistema.

Rootkit LKM opravi 3 naloge. Najprej premakne kodo hipervizorja v skriti pomnilnik. Nato namesti premaknjen hipervizor tako, da virtualizira delujoči operacijski sistem. Na koncu mora izolirati pomnilnik hipervizorja od pomnilnika gostujučega sistema s pomočjo Intelovega EPT.

Za samo virtualizacijo sistema je potrebno opraviti par korakov. Ali procesor ustrezza minimalnim zahtevam za VMX operacije, ki so nujne za namestitev virtualnega računalnika. Nato se inicializira hipervizorjeve podatkovne strukture. Kot je VMSC oz. struktura za nadzor virtualnega računalnika. Ta se uporablja tako za konfiguracijo hipervizorja kot gostujučega sistema. S tem lahko končno delujoči operacijski sistem virtualiziramo, tako, da prezrcalimo sistemo trenutno stanje v stanje gostujuče virtualne naprave. Na primer pod preslikavo spada stanje procesorja (registri). Na koncu VMCS uporabimo za registracijo določenih dogodkov, ki morajo biti prestreženi s strani hipervizorja ob delovanju.

5.2 Omejitve

Raziskovalci so Styx praktično preskusili le na 64-bitnem Linux operacijskem sistemu, z eno jedrnim Intelovim procesorjem. Program bi se sicer v teoriji dalo dopolniti za delovanje na več jedrnih procesorjih.

5.3 Zaznavanje

Če nas zanima ali je naš gostujuči sistem okužen, je dovolj, da preverimo, če OS teče v virtualiziranem načinu. Težava

```
00000000-00000fff : Reserved
00001000-0009ebff : System RAM
0009ec00-0009ffff : Reserved
000a0000-000bffff : PCI Bus 0000:00
000c0000-000c7fff : Video ROM
000ca000-000cafff : Adapter ROM
000cc000-000cffff : PCI Bus 0000:00
000d0000-000d3fff : PCI Bus 0000:00
000d4000-000d7fff : PCI Bus 0000:00
000d8000-000dbfff : PCI Bus 0000:00
000dc000-000fffff : Reserved
000f0000-000fffff : System ROM
00100000-bfecffff : System RAM
53000000-53c031d0 : Kernel code
53c031d1-5439a87f : Kernel data
54974000-54dfffff : Kernel bss
bfed0000-bfefefff : ACPI Tables
bfeff000-bfefffff : ACPI Non-volatile Storage
bff00000-bfffffff : System RAM
c0000000-febfffff : PCI Bus 0000:00
c0000000-c01fffff : PCI Bus 0000:03
c0208000-c020bfff : 0000:00:10.0
```

Slika 2: Primer vsebine */proc/iomem*

nastopi, ker v večini primerov danes vsa oblaćna infrastruktura temelji na virtualizaciji. V tem primeru je omenjeni test nesmiseln. Zato je edini učinkovit način zaznave, da poznamo pravo velikost fizičnega pomnilnika oziroma dodeljenega pomnilnika s strani hipervizorja (na primer vmware, virtualbox, itd.). Nato lahko primerjamo znano velikost pomnilnika s prebrano velikostjo. Če se velikosti ne ujemata je zelo velika verjetnost, da nekdo tretji upravlja z našim pomnilnikom (Styx v hipervizor načinu).

6. V PRAKSI

Praktične implementacije prototipa sistema Styx nismo izdelali, saj bi bil razvoj bistveno preveč zahteven in bi delo segalo izven okvirja seminarske naloge. Kljub temu smo se preko že implementiranega sistema za potrebe razumevanja delovanja Styx idejno sprehodili čez njegove osnovne komponente.

Spodaj so opisani nekateri ključni gradniki, ki bi jih prototip uporabljal pri delovanju.

V datoteki */proc/iomem* so zapisane preslikave sistemskoga pomnilnika na fizične naprave. Vsebino lahko izpišemo s preprostim ukazom *cat /proc/iomem*.

Če vemo, da operacijski sistem teče neposredno na strojni opremi, se lahko prepričamo, da nismo žrtev Styx s ukazom *cat /proc/cpuinfo | grep hypervisor*. Če v izpisu dobimo besedo hypervisor pomeni, da operacijski sistem teče v virtualiziranem načinu.

Izboljšave

Članek vsebuje tudi opis izboljšave orodja Rekall oz. pmem z dodajanjem algoritma za specifično iskanje sledi sistema Styx. Po uspešnem nalaganju sistema Styx je opisan tudi poskus preslikovanja pomnilnika s prilagojenima oz. do-

delanima orodjem Rekall in pmem. Orodji sta bili kljub modifikaciji nezmožni pridobiti uporabne slike pomnilnikov, kar nakazuje, da sistem Styx uspešno skriva pomnilnik, tudi pred najmodernejšimi orodji pomnilniške forenzike. Opazili smo tudi, da sta orodji bili nezmožni zaznati kakršnokoli sled izvajanja sistema Styx.

7. SLABOST

Glavno slabost sistema Styx predstavlja odvisnost sistema od količine skritega pomnilnika na digitalni napravi. Pomembna slabost prototipa sistema Styx leži tudi v dostopanju do pomnilnika z uporabo DMA oz. direct memory access (DMA dostopa do pomnilnika tako, da izogne centralni procesni enoti oz. procesorju), v primeru načina dostopa do pomnilniških vrednosti z uporabo DMA, bi tovrstno orodje hitro zaznalo izvajanje Styxovega rootkit-a, kar pa bi omogočilo analizo sistema, ter hitro razbitje skrivanja pomnilnika, ki ga omogoča sistem Styx. To ranljivost, lahko odpravimo z uporabo virtualiziranja DMA dostopa z Intelovo VT-d razširivijo ali pa z rekonfiguriranjem IOMMU (Input output memory management unit), da vse bralne ter pisalne zahtevke preusmerjamamo na enak način, kot je opisano zgoraj. S pomočjo te lahko hipervizor onemogoči DMA napravam dostop do specifičnega dela pomnilnika. Torej samo enostavna dopolnitev sistemu Styx omogoči skrivanje lastne prisotnosti.

8. ZAKLJUČEK

Opisan koncept sistema Styx prikazuje, da lahko tudi najpogostejsa orodja, kot sta npr. pmem in rekall, oslabimo z uporabo skritega pomnilnika ter t.i. "run-time" virtualizacije.

V članku je opisan način preslikave trenutno izvajajočega operacijskega sistema Linux v virtualizirano rešitev, podprt s strani strojne opreme računalnika, način pridobivanja podatkov LKM segmentov skupaj z algoritmom za oštevilčevanje skritega spomina ter način za prenašanje rootkita v skriti del pomnilnika, za potrebe zakrivanja sledi sistema.

Pri zakrivanju sledi delovanja Styxa uporabljamо tudi Intelov EPT preko katerega manipuliramo z MMU. Večji problem predstavljajo predvsem mnoge sledi, ki jih nalaganje Styxa pusti v pomnilniku, zaradi tega je v članku opisana tudi metoda sekundarnega LKM oz. Loadable kernel module, ki poskrbi za izbris teh sledi.

Prototip sistema Styx se lahko uporablja tako za dobre kot za zlonamerne namene. Pri postavitvi zlonamerne programske opreme na računalnik žrtve lahko napadalci izkoristijo sistem predvsem za skrivanje delovanja zlonamerne kode v računalniškem pomnilniku. To predstavlja resno grožnjo, saj je v takem primeru zlonamerni program skoraj nemogoče zaznati in odstraniti.

Zaradi specifičnosti uporabe sistema Styx in pa zaradi raznolikosti računalniških konfiguracij, kot sta količina skritega pomnilnika in različni načini implementacije virtualizacije pri različnih proizvajalcih procesorjev, je razširjena uporaba sistema Styx malo verjetna. Uporaba sistema je posledično bolj verjetna v manjšem številu primerov, ko napadalci oz. raziskovalci dobro poznajo konfiguracijo računalnika, ki ga želijo analizirati oz. napasti. Iz tega je moč sklepati, da so

alternativne metode pridobivanja vrednosti pomnilnika še vedno pomemben del digitalne forenzike.

9. REFERENCE

- [1] M. B. Athreya. *Subverting linux on-the-fly using hardware virtualization technology*. PhD thesis, Georgia Institute of Technology, 2010.
- [2] P. OKane, S. Sezer, and K. McLaughlin. Obfuscation: The hidden malware. *IEEE Security & Privacy*, 9(5):41–47, 2011.
- [3] R. Palutke and F. Freiling. Styx: Countering robust memory acquisition. *Digital Investigation*, 24:S18–S28, 2018.
- [4] J. Stüttgen and M. Cohen. Anti-forensic resilient memory acquisition. *Digital investigation*, 10:S105–S115, 2013.
- [5] J. Stüttgen and M. Cohen. Robust linux memory acquisition with minimal target impact. *Digital Investigation*, 11:S112–S119, 2014.

Nadzorovani poskusi pri ponaredbi forenzičnih dokazov

[Digitalna forenzika 2018/2019]

Domen Gašperlin
Fakulteta za računalništvo in
informatiko
Večna pot 113
1000, Ljubljana
dg6342@student.uni-lj.si

Domen Kajdič
Fakulteta za računalništvo in
informatiko
Večna pot 113
1000, Ljubljana
dk7480@student.uni-lj.si

Jaka Kerdež
Fakulteta za računalništvo in
informatiko
Večna pot 113
1000, Ljubljana
jk5456@student.uni-lj.si

POVZETEK

Število digitalnih dokazov narašča iz dneva v dan [3]. Če je nekdaj veljalo, da so bili na sodišču pretežno fizični dokazi, danes velja obratno [7]. Živimo namreč v digitalno dobi, kjer ima vsak posameznik dostop do več pametnih naprav. Digitalni dokazi prestavlajo določen izziv, saj jim ne moremo vedno zaupati. Te teme sta se dotaknila avtorja Felix Freiling in Leonhard Hösch [6], ki sta pri študentih digitalne forenzike naredila poskus ponaredbe digitalnih dokazov. V tem prispevku bomo njun poskus analizirali in predstavili ugotovitve. Gre za poskus manipulacije slike diska, pri čemer so morali študenti sliko diska spremeniti tako, kot da bi bil na njej opravljen dostop do spletne strani v preteklosti. Na koncu bomo tudi podali naše svoje mnenje o preizkusu, kaj bi morda dodali ali spremenili.

KLJUČNE BESEDE

antiforenzična, digitalna preiskava, ponarejanje dokazov, poskus

1. UVOD

Zaradi vse večjega srečevanja z digitalnimi dokazi, morajo biti forenzični preiskovalci posebej pozorni na njihovo avtentičnost. Med digitalnimi preiskovalci velja mešano prepričanje glede popolne ponaredbe dokazov. Večina jih verjame, da je to odvisno od znanja ter da večjo znanje prinese boljša ponaredbo (morda celo takšno, ki je ni mogoče zaznati) [2]. K temu dodatno pripomore digitalizacija, ki prinaša višjo izobraženost ljudi na področju računalništva. To lahko vodi v večje število ponarejenih digitalnih dokazov. Ponaredbe se lahko zgodi na dveh mestih: pred zasedbo (obtoženi izvede ponaredbo na svojem računalniku ali kakšni drugih napravi) ali med/po zasedbi dokazov na kraju zločina. Forenzični preiskovalec se lahko iz kakršnih kolih razlogov (podkupnine, grožnje...) odloči, da bo dokaze ponaredil. Poraja se vprašanje, kako težko je takšno manipulacijo dokazov zaznati in preprečiti, da ne bi prišlo do napačnih obtožb.

Da bi to preverila, sta avtorja načrtovala poskus pri študentih računalništva. Glavni cilj poskusa je bil ugotoviti, koliko dela je potrebno vložiti za dobro ponaredbo dokazov. Še posebej ju je zanimalo, v koliko primerih bodo študenti uspešni pri ponaredbi. Drugi cilj poskusa pa je bil ugotoviti, kaj vpliva na dobro ponaredbo. To sta preverjala na dva načina: s količino vloženega časa in stopnjo nadzora nad podatki (popoln dostop ali dostop na daljavo; pojasnjeno v delu *Predstavitev poskusa*). Poleg ponarejanja dokazov so študenti morali preveriti, kako dobro so se obnesli pri ponaredbi njihovi kolegi, tako da so analizirali njihovo dela. Čeprav je bil poskus izveden na majhnem vzorcu ljudi, lahko iz njega vseeno potegnemo zanimive ugotovitve, ki jih bomo predstavili v kasnejših sklopih.

2. PREGLED PODROČJA

Pred analizo članka smo pregledali področje antiforenzične. Globalno gledano z uporabo antiforenzične narašča tudi število orodij, ki so na voljo [4]. Kljub temu obstaja zelo malo akademskih raziskav na temu področju. Omenjen je celo podatek, da je le 2% člankov o digitalni forenziki namenjenih antiforenzični [1]. Posledično je področje zelo slabo formalno definirano. Ugotovili smo, da vsak ki piše o tem področju, antiforenzično definira na svoj način. Tako je na voljo veliko število različnih definicij, ki z leti postajajo kompleksnejše. Definicija iz leta 2002 pravi, da je antiforenzična samo skrivljanje poskusa vdora v sistem [8]. Novejša definicija iz leta 2012 pravi, da je antiforenzična skupek znanstvenih metod, ki so namenjene oteževanju dela forenzikom na vseh stopnjah preiskave [5]. Naš cilj ni primerjava različnih definicij, ampak ilustracija, da področje ni zelo akademsko aktivno in je šele v povojih. Zainteresirani bralec se lahko z več definicijami seznaniti v članku "Anti-forensics: Furthering digital forensic science through a new extended, granular taxonomy" [4].

Celovito gledano, je področje razdeljeno na štiri podpodročja [4]:

- skrivljanje podatkov,
- brisanje podatkov,
- zakrivljanje sledi,
- napadi na forenzična orodja ¹.

¹Področja napadov na forenzična orodja ni smiseln deliti na kategorije.

Podpodročja se delijo na več kategorij, omenili bomo samo popularnejše.

V področje skrivanja podatkov spadajo naslednje kategorije:

- šifriranje podatkov (diski, e-pošta, datoteke, gesla ...),
- steganografija (skrivanje podatkov znotraj druge datoteke; npr. skrivanje podatkov znotraj slike, znotraj besedila ...),
- druga skrivanja podatkov (v spominu računalnika, med participijami - angl. *slack* ...),
- skrivanje omrežnih podatkov (emulatorji terminalov, uporaba VPN - dostop do interneta preko tretje osebe).

V področje brisanja podatkov spadajo naslednje kategorije:

- brisanje datotek,
- brisanje diskov,
- brisanje odstranljivih pomnilniških medijev (npr. ključ USB),
- brisanje registrov,
- brisanje metapodatkov.

V področje zakrivanja sledi spadajo naslednje kategorije:

- omrežje P2P (angl. peer to peer; uporabljen za zakrivanje sledi; pogosto tudi v kriminalne namene [4]),
- zakrivanja naslova IP (angl. *spoofing*),
- namestniški strežniki (angl. *proxy servers*).

Zadnje področje (napadi na forenzična orodja) se sicer ne deli na kategorije, lahko pa ilustriramo primer takega napada. Napadalec bi lahko pogledal izvorno kodo odprtakodnega orodja in glede na njegovo delovanje oblikoval napad, zaradi katerega to orodje več ne bi pravilno delovalo.

Avtorja navajata, da je poskus, ki sta ga izvedla, nekaj novega na področju antiforenzike. Pravita, da kljub temu, da je bil le-ta izведен na majhnem vzorcu, lahko služi kot podlaga za nadaljnje študije na tem področju.

3. PREDSTAVITEV POSKUSA

Poskus so opravili na Friedrich-Alexander-Universität v nemškem Erlangnu. Sodelovalo je 14 študentov magistrskega programa računalništva. Naloga je bila predelati sliko diska (v nadaljevanju slika) tako, da bi forenzični preiskovalci ob njeni analizi ugotovili nedovoljen prenos datotek iz spleta. Vsak študent je dobil začetno sliko diska z nameščenim operacijskim sistemom Ubuntu. Cilj je bil, da je iz slike diska razviden prenos nemške strani iz Wikipedije za termin Rhioceros skupaj s fotografijami v obdobju 2.-17. 10. 2016.

Poskus je bil razdeljen na dva dela:

- v prvem so imeli študenti popoln nadzor nad sliko, kot bi ga imel napadalec v primeru neposrednega dostopa do računalnika,
- v drugem so simulirali napad na daljavo preko zlonamernega programa.

Rezultat prvega dela je bila zato ponarejena slika sama, pri drugem pa so študenti oddali program, ki se je pognal na originalni sliki.

Sledil je korak analize, kjer je študent z enako verjetnostjo dobil avtentično sliko ali ponaredek enega od ostalih študentov. Dobljeno sliko so analizirali s pomočjo orodij za forenzično preiskovanje. Na podlagi ugotovitev so morali podati mnenje, če je njihova slika ponarejena. Kakovost izdelanih ponaredkov je domnevno povezana s predznanjem vsakega študenta in vloženim trudom. Da bi eksperiment upošteval tudi te razlike, so morali sodelujoči na začetku izpolniti vprašalnik o predznanju, med samim delom pa voditi dnevnik o opravljenem delu in porabljenem času.

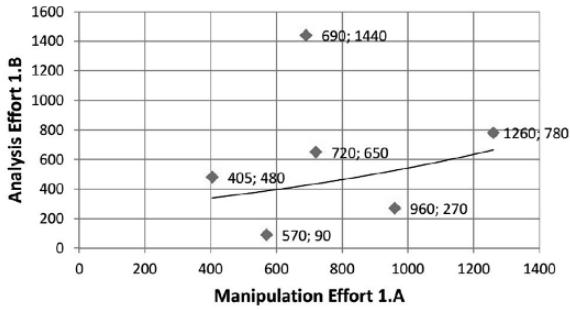
4. REZULTATI POSKUSA

V tem poglavju so opisani izsledki raziskave. Predstavljene so glavne ugotovitve, podrobno so opisani tudi pristopi posameznih študentov pri ponarejanju. Na koncu pa so predstavljeni naši sklepi in komentarji.

Pregled začetnih vprašalnikov je pokazal, da imajo vsi študenti podobno predznanje in so motivirani za sodelovanje pri eksperimentu. Vsi so namreč zaključili uvod v forenziko, kjer so spoznali osnove analize diskov. Večina študentov je ta predmet zaključila z dobro oceno in pokazala motivacijo za opravljanje eksperimenta. Na teden so bili pripravljeni vložiti 20-30% časa namenjenega študijskih obveznostim. Iz teh podatkov avtorja sklepata, da so rezultati med seboj primerljivi.

4.1 Kvantitativni rezultati

Dnevniki opravljenega dela so pokazali, da so študenti za izdelavo ponaredka porabili različno časa. V nadaljevanju zato avtorja primerjata korelacijo med časom ponarejanja in časom analize ponarejene slike. V prvem delu je sodelovalo 11 študentov, od tega pa jih je v analizo dobilo ponaredek le 6. V drugem delu je sodelovalo 7 študentov, le trije od njih pa so prejeli ponaredek. Program študenta #11 za ponarejanje slike se žal ni izvedel, zato je bila slika nespremenjena. Ta primer je bil odstranjen iz končnih rezultatov. Prejete slike diskov in časi za prvi del so predstavljeni na sliki 1 in v tabeli 1. Na enak način so na sliki 2 in v tabeli 2 predstavljeni podatki za drugi del. Dve analizirani sliki v drugem delu sta premalo, da bi lahko pokazali naraščajočo odvisnost, kakršna je razvidna v prvem. Avtorja sta pričakovano ugotovila, da več vloženega truda v izdelavo ponaredka zahteva tudi več časa za njegovo dokazovanje. Te ugotovitve žal nista podkrepila z dokazom, saj članek navaja vložen trud le v tri analizirane ponaredke. Poleg tega lahko opazimo še veliko razliko v povprečnem času med analizo originala (386 minut v povprečju) in analizo ponaredka (618 minut v povprečju). Iz tega lahko sklepamo to, da je pri analizi



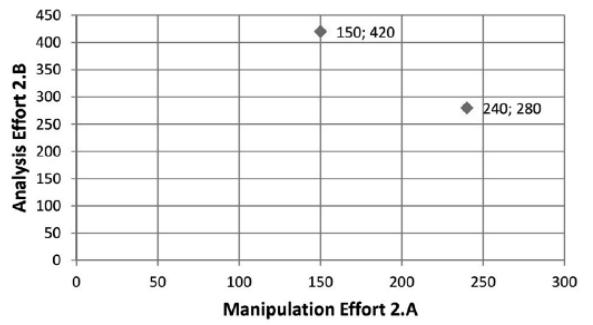
Slika 1: Vsaka točka na grafu prikazuje potreben čas za izdelavo (X os) in analizo (Y os) ponaredka v prvem delu (neposredni dostop).

Tabela 1: Za vsakega sodelujočega v prvem delu je zabeležena prejeta slika, čas njene izdelave in analize v minutah in pravilnost klasifikacije.

Sodelujoči	Prejeta	Izdelava	Analiza	Pravilnost
1	Original	-	855	Da
2	Original	-	270	Da
3	11	405	480	Da
5	6	720	650	Da
6	Original	-	180	Da
7	Original	-	240	Da
8	13	690	1440	Da
9	8	960	270	Da
11	1	570	90	Da
12	3	1260	780	Da
13	Original	-	900	Ne

ponaredkov potrebna večja pozornost, saj so manipulacije nepredvidljive. Za vsako morebitno odstopanje obstaja več različnih vzrokov. To pomeni, da vsako odstopanje morda ni manipulacija, ampak posledica nekih drugih faktorjev.

Vsi izdelani ponaredki za prvi in za drugi del, ki so šli skozi analizo so bili pravilno klasificirani. Napačna klasifikacija se je zgodila le v primeru, ko je študent v prvem delu dobil avtentično sliko in jo razglasil za ponaredek. En primer iz drugega dela je bil izvzet iz rezultatov, saj je pri izvajanju programa prišlo do napake. Ugotovljena je bila tudi korelacija med količino nadzora nad diskom in časi manipulacije/analize. Če ima napadalec popoln nadzor nad diskom (prvi del poskusa), je manipulacija hitrejša in zaznavanje počasnejše. Obratno velja pri manj nadzora. Takrat je manipulacije počasnejša, zaznavanje pa hitrejše. To lahko pripisemo temu, da mora napadalec brez popolnega dostopa do slike veliko bolj premisliti, kaj bo naredil, saj ima zelo malo konkretnih informacij o dejanski sliki. Njegova manipulacija je takrat striktno programska, kar pomeni, da je že samo pisanje programa zamudno. Krajši čas zaznavanja manipulacije pa pripisemo temu, da napadalec ne more predvideti vseh faktorjev in tako laže stori napako pri ponaredbi.



Slika 2: Vsaka točka na grafu prikazuje potreben čas za izdelavo (X os) in analizo (Y os) ponaredka v drugem delu (posredni dostop).

Tabela 2: Za vsakega sodelujočega v drugem delu je zapisana prejeta slika, čas njene izdelave in analize v minutah in pravilnost klasifikacije.

Sodelujoči	Prejeta	Izdelava	Analiza	Pravilnost
1	Original	-	186	Da
3	11	840	270	Da ¹
5	9	150	420	Da
7	Original	-	240	Da
11	Original	-	90	Da
12	Original	-	720	Da
13	12	240	280	Da

¹ Ta primer je bil odstranjen iz končnih rezultatov, saj se program za izdelavo ponaredka ni izvedel.

4.2 Kvalitativni rezultati

Zanimivi so tudi kvalitativni rezultati, ki prikazujejo več različnih načinov, s katerimi so študenti poskušali izdelati ponaredke. Za nekaj študentov bomo te načine prikazali in argumentirali kaj so naredili narobe (ali morda spregledali), da je bila njihova slika diska klasificirana kot ponaredek.

4.2.1 Študent #1

Študent #1 se je ponaredbe lotil tako, da je najprej prižgal sliko (angl. boot) in dostopal do Wikipedije. Fotografije nosorogov je prenesel, kot bi to naredil avtentičen uporabnik. Nato je poskušal zakriti svoje sledi s spremembou časovnih žigov, ki so bili pri tem dostopu ustvarjeni (časi MAC prenesenih fotografij; zgodovina brskalnika v datoteki *places.sqlite* in dnevniške datoteke *syslog*, *auth.log* in *bash history*). V dnevnik opravljenega dela je bilo zapisano, da storitve NTP (angl. *Network Time Protocol*) študentu ni uspelo izklopiti. Študent je tako zaključil, da so bili časovni žigi spremenjeni znotraj prižganega sistema.

Sliko študenta #1 je analiziral študent #11. Ta je znotraj slike ugotovil veliko neskladnosti. Najbolj očitna pomankljivost so bili napačni časovni žigi v brskalniku (datoteki *places.sqlite* in *cookies.sqlite*), v nadzorniku aktivnosti *Zeitgeist* (datoteka *activity.sqlite*) in v seznamu zadnjih odprtih datotek v sistemu Ubuntu. Poleg tega je študent spregledal še veliko drugih stvari. Pozabil je na datoteko *auth.log*, ki je prikazovala tri prijave v sistem ob napačnem času. V času dostopa do slike je študent namestil paket (in pozabil spremeniti časovne žige), kar je spet nakazovalo na manip-

ulacijo. Zadnja odkrita pomanjkljivost je bila zaznana v zgodovini ukazne lupine Bash. Študent je namreč pozabil izbrisati zgodovino izvedenih ukazov v datoteki *bash_history*. Iz te datoteke je bilo precej jasno, kakšne ukaze je študent izvajal, ko je poskušal izvesti manipulacijo. Študent #11 je lahko brez problema zaključil, da je ta slika ponaredek zaradi velikih neskladnosti.

4.2.2 Študent #3

Študent #3 se je ponaredbe lotil podobno kot študent #1. Najprej je prižgal sliko in naredil kopijo vseh pomembnih dnevniških datotek. Nato je določil realistične časovne žige dostopov do Wikipedije in prenosa fotografij glede na čas zadnje prijave v sistem ter dostope opravil. Prvotni načrt je bil premik sistemskih ure v preteklost ter obisk Wikipedije, vendar bi takšen pristop pripeljal do problemov s spletnimi certifikati. Tako so študentu ostale dvojne dnevniške datoteke: prvotne in nove z dodanimi vnosni dostopovi. Naslednji korak je bil izklop sinhronizacije časa in premik ure nazaj zaradi dodajanja novih vnosov v prvotne dnevniške datoteke (glede na novo pridobljene dnevniške datoteke). Študent je za ta korak napisal program zaradi lažjih sprememb večjega števila datotek. Še posebej je bil pozoren pri spremembah zgodovine obiskanih spletnih strani (*places.sqlite*), kjer je poskrbel, da se primarni ključi v bazi *Sqlite* ujemajo. Na koncu je izbrisal nepotrebne dnevniške datoteke in vstavljal prvotno zgodovino ukazne lupine (*bash_history*) ter poskrbel, da so časi MAC v vseh spremenjenih datotekah nastavljeni na enake čase, kot so bili pred spremembom.

Sliko študenta #3 je analiziral študent #12. Ugotovil je, da so datoteke spletnega brskalnika konsistentne. Kljub temu pa je naletel na neskladnosti pri analizi dnevniških datotek, saj so nekatere izmed njih bile prazne (*wtmp* in *auth.log*). Očitno ponovno vstavljanje dnevniških datotek ni delovalo pravilno. Podobno tudi časi MAC prenesenih fotografij niso bili ustrezni. Usodna napaka je bila pomanjkljiva obdelava datoteke *bash_history*, ki je vsebovala ukaze za manipulacijo sistemskega časa.

4.2.3 Študent #6

Študent #6 je svojo ponaredbo začel z ugotavljanjem primernega časa dostopa glede na prejšnje čase prijav v sistem (datoteka */var/log/wtmp*). Nato je sliko prižgal, obiskal Wikipedijo in prenesel zahtevani fotografiji. Ves čas je beležil, katere datoteke so se med tem na disku spremenile in ali so bile morda na disk zapisane kakšne nove datoteke. Te datoteke je nato analiziral zunaj slike in se lotil njihove manipulacije. Najprej je napisal program, ki spreminja časovne žige znotraj predpomnilniških datotek (format *Cache2*) v brskalniku Firefox. Časovne žige znotraj podatkovnih baz SQLite je spremenil ročno s pomočjo ukazov SQL. Na koncu je spremenjene datoteke vstavil nazaj v originalni disk ter poskrbel, da so časovni žigi teh datotek primerni (časovne žige *inode* je spremenil s pomočjo skripte in orodja *debugfs*).

Sliko študenta #6 je analiziral študent #5. Znotraj predpomnilniških datotek in zgodovine spletnega brskalnika Firefox ni našel nobenih očitnih problemov. Je pa ugotovil, da vnos ene izmed dveh zahtevanih fotografij manjka v zgodovini brskalnika in da so bili časi MAC prenesenih datotek nekonsistentni s časi prenosov v zgodovini brskalnika. To je bilo dovolj, da je sliko klasificiral kot ponaredek.

4.2.4 Študent #8

Študent #8 je poskrbel za varnostno kopijo vseh relevantnih datotek in zatem prižgal sliko ter obiskal Wikipedijo. Po zaustavitvi slike je študent primerjal razlike med predhodnimi in na novo nastalimi datotekami. S tem znanjem je posodobil dnevniške datoteke s ponarejenimi časi prenosov datotek. Kar nekaj truda je bilo vloženega v spremembo SQLite baze z zgodovino brskanja in nastavljivo ustreznih časovnih žigov MAC. Vse kasnejše manipulacije so bile narejene s priklopom slike (angl. mount).

Sliko študenta #8 je analiziral študent #9. Pri svoji analizi je ugotovil, da je slika vsebovala prijave v sistem po času dostopa do Wikipedije. Poleg tega so bili v datoteki *places.sqlite* prisotni nekonsistentni časovni žigi. S tem je ugotovil, da gre za ponaredbo.

4.2.5 Študent #11

Študent #11 se je sprva ponaredbe lotil s spremembami sistemskega časa. Naletel je na težavo, saj mu je zaradi preverjanja certifikata strežnik Wikipedije preprečil dostop. Nato se je lotil ročne manipulacije vseh potrebnih datotek na sliki diska kot so piškotki, zgodovina brskanja, datoteka *activity.sqlite*, nedavno uporabljeni datoteke v Ubuntu Dash-u, časi MAC in datoteke nadzornika aktivnosti Zeitgeist.

Sliko študenta #11 je analiziral študent #3, ki je ugotovil, da so časi MAC nekonsistentni z zgodovino prenosov. Npr. čas kreacije datoteke je bil po času spremembe in času dostopa. Študent je prišel do pravilnega zaključka, da je šlo za manipulacijo.

4.2.6 Študent #13

Študent #13 se je ponaredbe lotil drugega kot ostali. Najprej je prižgal sliko, opravil dostope, prenesel fotografije in nato izbrisal vse pomembne datoteke (zgodovino ukazov lupine *bash_history*, prenesene fotografije, zgodovino brskalnika ...). Zgodovina brskalnika je bila izbrisana direktno z brskalnikom. S tem je študent preprečil pridobivanje zgodovine s pomočjo t.i. *carvinga*. Svojo manipulacijo diska je predstavil kot zgodbo, da se je uporabnik pogovarjal s prodajalcem nosorogov na črnem trgu, ki je zraven priskrbel še navodila, kako se izbriše vse dokaze. Da bi bila zgodba bolj verjetna, je študent dodal še pogovor z neko osebo, ki je vseboval podrobnosti o oddaljenem dostopu do diska. Ta ista oseba je potem podtaknila škodljivo kodo (*ZIP bomb*), ki ob odprtju doda ukaz *rm -rf* v datoteko *.bashrc* trenutnega uporabnika.

Sliko študenta #13 je analiziral študent #8. Pri tem je naletel na težave, saj ni točno vedel, kaj se je zgodilo. Ko je pregledoval izbrisane datoteke, je lahko obnovil le imena datotek in ne njihove vsebine. Je pa odkril protokol za pogovor, dokaze o zunanjem dostopu do slike kot tudi spremembe sistemskega časa v datoteki *auth.log*. Poleg tega je naletel še na škodljivo kodo (*ZIP bomb*) in na zlonamerni ukaz *rm -rf* v datoteki *.bashrc* in nekonsistentne časovne žige. Tako je študent zaključil, da so bili dokazi nekonsistentni in da je manipulacija zelo verjetna.

4.2.7 Celovit pogled na kvalitativne rezultate

Iz zgornjih rezultatov lahko povzamemo tri sklepe. Prvi sklep je, da več vloženega truda v ponaredbo pomeni težje zaznavanje, če je disk ponaredek. To je bilo še posebej vidno pri študentu #13, ki je v ponarejanje vložil največ dela.

Drugi sklep je precej logičen, saj nakazuje na to, da že najmanjsa napaka lahko hitro izda ali je disk ponaredek. To je posebej razvidno pri študentih #1 in #8, ki sta naredila očitne napake. V teh primerih je bilo odkrivanje precej kraje (90 in 270 minut). Prav tako se je pojavilo odstopanje. Študent #12 je porabil skupno 780 minut za analizo slike študenta #3, čeprav je ta vsebovala očitne napake.

Zadnji sklep se nanaša na tip manipulacije. Pravi namreč, da manipulacija zunaj slike vodi k bolj prepričljivim ponaredkom (v primeru polnega dostopa; se nanaša na priklop slike in ne na prižig). Na ta sklep lahko pogledamo iz logične perspektive: že sama prijava v sistem ustvari veliko število dokazov. Skoraj vse, kar se dogaja na računalniku se na nek način beleži. Če k temu dodamo še omrežne dostope, se vse dodatno zaplete. Omrežne interakcije so namreč neterminate in neponovljive.

5. ZAKLJUČEK

Spoznali smo, da ponarejanje digitalnih dokazov ni lahka naloga. Trud, ki ga moramo vložiti za izdelavo ponaredka, je veliko večji, kot pa da ponaredek odkrijemo. Velik vpliv ima tudi stopnja dostopa, ki jo ima napadalec na voljo nad računalnikom. Eksperimenti so pokazali, da manjša stopnja dostopa olajša delo preiskovalcu in otežuje delo ponarejevalcu.

Po našem mnenju bi moral poskus potekati v bolj nadzorovanem okolju nad množico prostovoljcev, ki se med seboj ne poznajo. Le tako bi se izognili drugim faktorjem kot so študijske obveznosti, nepristranskost in morebitno sodelovanje študentov, ki lahko vpliva na kakovost opravljenega dela. Tako bi sodelujoči prihajali iz različnih univerz in okolij. Imeli bi različna znanja in izkušnje. Potrebno bi bilo povečati tudi število sodelujočih, da bi lahko dobili statistično značilen vzorec populacije.

Prenove bi lahko bile deležne tudi metrike za merjenje vloženega truda. Lahko bi jih razširili z upoštevanjem predznanja in počutja udeležencev. Čas potreben za razrešitev primera namreč zadostno ne zajeme vseh dejavnikov. Nekatere od teh sta avtorja zajela v vprašalniku pred začetkom eksperimenta, vendar jih v delu nista predstavila kljub temu, da so bili študenti med seboj po njunem mnenju primerljivi. Predlagamo, da bi dejavnike, ki vplivajo na potek forenzične preiskave predhodno preučila in hipoteze o tem potrdila oz. ovrgla.

V kolikor bi se želeli prepričati o učinkovitosti in težavnosti antiforenzi, bi morali v raziskavo vključiti še forenzične preiskovalce. Ti imajo na tem področju več znanja in izkušenj kot študenti. Če bi imeli tudi oni težave s podtikanjem dokazov, potem bi bil to močen argument proti antiforenzi.

6. VIRI

- [1] I. Baggili and et al. Research trends in digital forensic science: an empirical analysis of published research. 2012.
- [2] M. Caloyannides. Digital “evidence” and reasonable doubt. 2003.
- [3] E. Casey. Digital evidence and computer crime. 2011.
- [4] K. Conlan, I. Baggili, and F. Breitinger. Anti-forensics: Furthering digital forensic science through a new extended, granular taxonomy. 2016.
- [5] K. Dahbur and B. Mohammad. Toward understanding the challenges and countermeasures in computer anti-forensics. 2012.
- [6] F. Freiling and L. Hosch. Controlled experiments in digital evidence tampering. 2018.
- [7] P. Kirk. Crime investigation. 1974.
- [8] B. Shirani. Anti-forensics. 2002.

Forenzika poškodovanih digitalnih naprav

Jaka Krajnc
Faculty of Computer and
Information Science
University of Ljubljana
jk8279@student.uni-lj.si

Nejc Rebernik
Faculty of Computer and
Information Science
University of Ljubljana
nr3054@student.uni-lj.si

POVZETEK

V zadnjem času je uporaba elektronskih naprav postala del našega vsakdana. V okviru digitalne preiskave lahko iz elektronskih naprav osumljenca pridobimo določene podatke, ki so lahko del dokaznega gradiva na sodišču. Zgodijo pa se primeri, ko je bila naprava ki jo dobimo v obravnavo podvržena določenim fizičnim poškodbam. Kljub poškodbam naprave želimo iz nje pridobiti uporabne podatke. Ker se poškodovane naprave ne znajdejo tako pogosto v forenzičnih preiskavah, še ni oblikovanega strokovnega znanja, kako obravnavati poškodbe naprave. Prav tako na to temo še ni bilo naslovljeno veliko raziskav. V članku bomo opisali glavne vrste poškodb na digitalnih napravah, ter kako postopati v primeru obravnave poškodovane naprave. Pregledali bomo tudi načine pridobitve podatkov iz poškodovanih mobilnih naprav, ter možnosti pridobitve podatkov iz poškodovanih pomnilniških medijev.

Ključne besede

Poškodovane naprave, tipi poškodb, forenzika pomnilnika, pridobivanje podatkov

1. UVOD

Po celiem svetu obstajajo poškodovane naprave, ki čakajo, da nekdo iz njih prikaže podatke potrebne za preiskave. Te naprave morda vsebujejo podatke, ki lahko razrešijo kazniva dejanja ali imena tistih, ki so bili napačno obtoženi, vendar se podatki štejejo za nedostopne ali pa je naprava uničena do te mere, da iz nje ni več moč pridobiti ustreznih podatkov. Naprave, ki delujejo tako kot so bile prvotno zasnovane pogosto niso združljive z orodji za pridobitev podatkov iz naprav, zato se štejejo kot nedostopne. Poškodbe na napravah so različne, lahko jih povzročijo različni naravnji faktorji in s tem onemogočijo funkcionalnost naprave ter rapoložljivost podatkov. Na napravah so lahko prisotne različne vrste poškodb [3] [1].

2. TIPI POŠKODB

V nadaljevanju bomo predstavili tipe poškodb, ki se lahko pojavijo na digitalnih napravah. Poškodbe elektronskih naprav tako lahko tipično delimo v naslednje kategorije [1].

- Poškodba s tekočino
- Termalna poškodba
- Balistična poškodba
- Poškodba zaradi udarca

Za vsako kategorijo je bila v okviru raziskave [1] izvedena simulacija škode na podoben način kot v resničnem svetu. Na podlagi simulacije so ocenili učinek poškodb na napravah. Za merjenje učinkov so bili uporabljeni znanstveni instrumenti in protokoli. Glede na stanje naprave po simulaciji so bile tako ugotovljene različne stopnje poškodovanosti naprave. Glede na stopnjo poškodovanosti naprave bodo določene metode, ki jih lahko uporabimo za pridobitev podatkov iz naprave. Cilj raziskave je bil tudi identifikacija katastrofalne škode na napravah, kar pomeni, da iz naprave več ni možno pridobiti nobenih podatkov. Prednosti izvedenih postopkov so osnovne znanstvene raziskave o vseh izčrpnih metodah za pridobivanje podatkov iz poškodovanih naprav. Cilj je tudi definirati protokole, ki jih je potrebno razviti in deliti za uspešno obravnavo poškodovanih naprav glede na način poškodovanosti. S protokoli želijo podpreti organe pregona da vključijo protokole za obravnavo poškodovanih naprav v obstoječe operativne postopek.

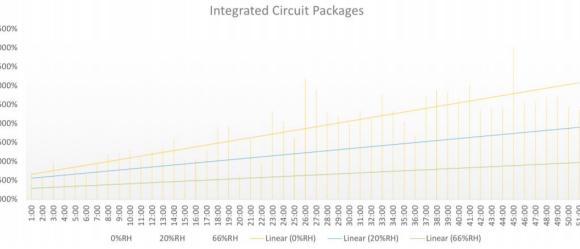
Vsek način škode vključuje vrsto individualnih študij ki se med seboj dopolnjujejo. Vse študije so osredotočene na izvedljivost pridobivanja podatkov iz poškodovanih naprav. Obstojče študije v panogi so v najboljšem primeru anekdotične in ne temeljijo na znanosti. V številnih panogah že obstajajo različni protokoli vendar niso bili usklajeni za uporabo pri pridobivanju podatkov za potrebe digitalne forenzike [1].

2.1 Poškodba s tekočino

Eden od tipov poškodb na napravah je poškodba s tekočino. Pri takšnih vrstah poškodbah pride do izraza zmogljivost integriranih vezij na napravi, da še prenesejo določeno obremenitev. Prav tako pomemben faktor šteje čas izpostavljenosti naprave v tekočini ter faktor izpostavljenosti naprave na določeni globini. Pri napravi, ki je bila podvržena poškodbi s tekočino je pomembna učinkovitost tehnik čiščenja

morebitnih nečistoč, prav tako pa je pomembno da napravo dobro osušimo (tehnike sušenja). Ker lahko gre pri poškodbah s tekočino tudi za vnetljive in biološko sporne tekočine je potrebno opredeliti biološke nevarnosti ter upoštevati morebitno vnetljivost tekočine.

V okviru preiskave poškodovanih naprav nas zanima koliko tekočine lahko v določenem časovnem obdobju obsorbirajo naprave glede na relativno vlažnost okolja, ter, če lahko dolčimo najvišjo stopnjo absorbcije tekočine. Izkaže se, da se skozi čas absorbcija vode v napravo povečuje, zato je pomembno da se tega zavedamo kadar imamo opravka z napravo, ki je bila izpostavljena poškodbi s tekočino.



Slika 1. Delež absorbcije vode skozi čas v napravo glede na relativno vlažnost okolja

Pri preiskavah naprav v digitalni forenziki poškodba s sladko vodo predstavlja največji delež poškodb s tekočino, nato ji sledi slana voda. Prav tako niso izjema poškodbe s krvjo žrtve ali napadalca. Povprečni čas, ko je bila naprava izpostavljena vodi znaša tri dni, prav tako povprečni čas do prvega čiščenja. Identificirani so bili ukrepi ki jih je potrebno izvesti, kadar se srečamo z elektronsko napravo, ki je bila izpostavljena poškodbi s tekočino so naslednji: [3]:

- Čimprej odstranimo baterijo
- Naprave ne prižigamo ali polnimo dokler se popolnoma ne osuši
- Napravo transportiramo v tekočini do laboratorija, kjer se opravi čiščenje
- Če transport v tekočini ni mogoč pred transportom napravo dobro osušimo
- Iz naprav je potrebno v primerem času pridobiti podatke in ne čakati predolgo

Izkaže se, da je zelo pomembno da se naprave izpostavljene tekočinam obravnavajo čimprej, saj ima izpostavljenos velik vpliv na stopnjo poškodovanosti naprave. Naprave je potrebno prevažati v tekočini od prizorišča do laboratorija kjer jih obravnavamo. Razlog za transport v tekočini je, da se ob osušitvi ustrezno ostrani morebitne usedline in trde delce. Prav tako je potrebno mokre naprave pred pričetkom analize temeljito osušiti [3].

2.2 Temperaturne poškodbe

Temperaturne poškodbe lahko opredelimo kot poškodbe, ki jih povzroči termična poškodba, mikrovalovne poškodbe ter



Slika 2. Poškodba s tekočino



Slika 3. Temperaturna poškodba

potencialna škoda, ki nastane zaradi pene, ki se uporablja pri gašenju požara [1].

Izkaže se, da so naprave veliko bolj odporne, kot si predstavljamo. Podatki so bili uspešno pridobljeni iz naprav, ki so prestale od 900 do 1000 stopinj termalne obremenitve. Pri termalnih poškodbah lahko pričakujemo izzive z dostopom in odpiranjem naprav, saj so te tipično že delno stopljene. Prav tako lahko pričakujemo da bodo čipi že delno poškodovani [1].

2.3 Balistične poškodbe

Pri balističnih poškodbah obravnavamo poškodbe z orožjem ter poškodbe zaradi eksploziva. Težave pri takšnih poškodbah so, če gre izstrelek direktno skozi pomnilnik, saj takrat pridobitev uporabnih podatkov praktično ni več možna, ker je bil s tem uničen pomnilnik. Če pa je izstrelek zadel v kakšen drug del naprave, se verjetnost pridobitve podatkov bistveno poveča [1].

Pri transportu v laboratorij in med obravnavo moramo biti posebej pozorni, da se naprava dodatno ne zdrobi ali zlomi. Vsako drobljenje in lomljenje lahko dodatno poškoduje komponente in oteži ali celo onemogoči obnovitev podatkov.

Poškodbe zaradi eksploziva lahko na napravi pustijo sledi eksplozivnih ali vnetljivih snovi. Pri desoldiranju lahko zaradi povišane temperature pride do vžiga ali celo eksplozije ostankov teh snovi. Čiščenje naprave je v takšnih primerih

posebej pomembno.

2.4 Poškodbe zaradi udarca

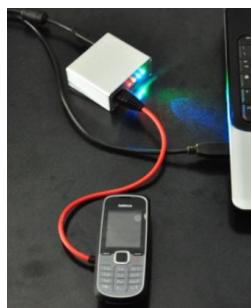
Poškodbam zaradi udarca so najpogosteje podvrženi mobilni telefoni ter tablični računalniki. Dobra plat takšnih poškodb je, da so podatki na napravah še vedno tipično nepoškodovani, kljub temu, da je telefon poškodovan in jih lahko najdemo v SIM karticah in ostalih pomnilniških karticah [1].

Pomnilniške naprave, ki uporabljajo pomnilnik Flash, kot je USB služijo za prenos podatkov iz točke A v točko B. V to kategorijo spadajo tudi kartice SD, ki se običajno nahajajo v skoraj vsaki napravi, od mobilnih telefonov in tabličnih računalnikov, do digitalnih fotoaparatorov, kot oblika izmenljivega shranjevanja podatkov. Ker so te naprave zelo majhne in krhke imajo večjo možnost, da jih poškodujemo do te mere da iz njih ne moremo več na enostaven način pridobiti pomembnih podatkov. Kadar prenosni medij več ne funkcioniра normalno takrat potrebujemo strokovnjaka na področju elektrotehnike, ki lahko preveri možnosti pridobitve podatkov iz poškodovane naprave.

V tej kategoriji lahko omenimo tudi poškodbe trdih diskov, ki že desetletja obvladujejo shranjevanje podatkov in so še vedno pri veliki večini ljudi glavna oblika shranjevanja podatkov. Ko se trdi disk pokvari potrebujemo posebna orodja in strokovno znanje za reševanje podatkov in nadaljnjo forenzično analizo. Nekaj metod pridobitev podatkov bomo opisali v nadaljevanju.

3. PRIDOBITEV PODATKOV IZ PAMETNIH TELEFONOV

Danes pametne telefone najdemo povsod, ti pa vsebujejo veliko informacij o lastniku in njegovih aktivnostih. Zaradi njihove velike razširjenosti bodo z veliko verjetnostjo vključeni tudi v kriminalna dejanja. Cilj digitalne forenzy mobilnih oziroma pametnih telefonov je povrniti potencialne digitalne dokaze tako, da bodo lahko kasneje predstavljeni sodišču. Obstaja več metod za pridobivanje dokazov iz mobilnih naprav vendar moramo v našem primeru upoštevati stopnjo poškodovanosti naprave. V splošnem poznamo več načinov pridobivanja podatkov iz poškodovanih mobilnih naprav, za katero se odločimo je odvisno tudi od poškodovanosti naprave [4].



Slika 4. Naprava Flasher Box

3.1 Ročna pridobitev podatkov

To je naljažja metoda pridobitve podatkov iz mobilne naprave. Preiskovalec uporabi tipkovnico telefona, da pridobi

želeno vsebino. Prednost te metode je, da je najlažja in ne zahteva poglobljenega znanja kako pridobiti podatke, deluje na vseh telefonih in ni potrebe po dodatnih kablih. Slabost metode pa je, da ne ohrani integritete dokaznega građiva, prav tako ni mogoče pridobiti podatkov o izbrisanih ali skritih datotekah. Metoda večinoma ne bo primerna, kadar imamo opravka s poškodovano napravo.

3.2 Logična pridobitev

Logična pridobitev podatkov pomeni pridobivanje podatkov, ki se nahajajo v logični participiji pomnilnika mobilnega telefona. Tako logična pridobitev ne pridobi podatkov, ki se nahajajo zunaj logičnih particij, kot je nedodeljen prostor. Ta postopek se izvede s povezavo telefona in računalnika preko infrardeče povezave ali povezave bluetooth, nato pa se izvedejo ustrezni ukazi za pridobitev podatkov iz naprave. V kolikor gre za hujšo poškodbo naprave tudi ta metoda ne bo primerna za pridobitev podatkov iz poškodovane mobilne naprave.

3.3 Fizična pridobitev

Fizična pridobitev podatkov je opredeljena kot kopiranje celotnega fizičnega polnilnika telefona. Za takšno pridobivanje podatkov uporabljam orodje Flasher box. To metodo lahko že s pridom uporabimo za potrebe pridobivanja podatkov, kadar imamo opravka s poškodovano mobilno napravo.

3.3.1 Flasher Box

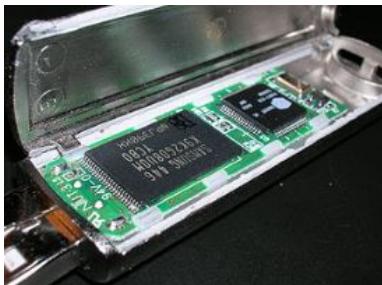
Orodje je sestavljeno iz programske opreme, gonilnikov in strojne opreme, ki izvaja pridobivanje podatkov, ki so shranjeni v pomnilniku mobilnih telefonov. Zajemanje podatkov na nižjem nivoju pomeni mimo operacijskega sistema in odpiranje komunikacijskih vrat za potrebo pridobitve vsebine podatkov v pomnilniku. Orodje Flasher box se ne uporablja samo za potrebe forenzy, ampak se lahko uporabi tudi za diagnostiko mobilnih naprav. Kot zanimivost lahko omenimo, da lahko s pomočjo omenjenega orodja sprememimo tudi IMEI identifikator telefona, mogoče pa so tudi posodobitve vdelane programske opreme ter konfiguracija različnih nastavitev mobilne naprave.

3.4 Pridobivanje podatkov iz čipa (angl. *Chip-off*)

Namen te metode je pridobivanje podatkov iz notranjega pomnilnika naprave. Postopek se izvede tako, da se pomnilnik odstrani iz vezja, nato očisti stike ter nakoncu prebere vsebina pomnilnika. Obstaja več metod za odstranitev pomnilnika iz vezja, metode so odvisne od tipa pomnilnika. Omenjena metoda ima kar nekaj prednosti. Metoda nam omogoča delo z vsemi vrstami mobilnih telefonov, prav tako lahko pridobimo celotno vsebino pomnilnika. Podatke iz pomnilnika pridobimo brez da bi storili kakršnokoli spremembo v obstoječih podatkih. Metoda torej ohranja celovitost podatkov. Poleg tega pa jo lahko uporabimo pri poškodovanih mobilnih napravah in lahko obide varnostno kodo. Slabost metode je, da lahko ob odstranjevanju pomnilnika poškodujemo čip in s tem izgubimo določen del dokazov.

4. PRIDOBIVANJE PODATKOV IZ POŠKODOVANIH USB FLASH POMNILNIKOV

Flash pomnilnik je vrsta računalniškega pomnilnika, ki ga lahko elektronsko izbrišemo ali reprogramiramo. Ta tehnologija je bila primarno uporabljena v pomnilniških karticah in USB ključih, ki so služili kot ključne naprave za prenos podatkov med računalniki in drugimi digitalnimi napravami. Flash pomnilnik je zvrst EEPROM-a [2]. Fizična poškodba flash pomnilnika ne bo odstranila naboja, ki je shranjen na tranzistorjih s plavajočimi vrti. Če je tranzistor še vedno nedotaknjen je mogoče prebrati njegove podatke (bit), vendar postopek v tem primeru ne bo enostaven. Dele flash pomnilnika lahko preberemo z laserskimi tehnikami in tehnikami merjenja žarkov (angl. *beam probing techniques*) [5].



Slika 5. USB Flash pomnilnik

- [2] The fundamentals of flash memory storage.
<https://www.electronicdesign.com/memory/fundamentals-flash-memory-storage>. Dostopno: 2.6.2019.
 - [3] Water damaged devices - evidence locker corrosion.
<https://www.slideshare.net/watsonsteve/ceic2015-watson-wdd001final>. Dostopno: 13.4.2019.
 - [4] K. A. Alghaffi, A. Jones, and T. A. Martin. Forensics data acquisition methods for mobile phones. In *2012 International Conference for Internet Technology and Secured Transactions*, pages 265–269. IEEE, 2012.
 - [5] B. J. Phillips, C. D. Schmidt, and D. R. Kelly. Recovering data from usb flash memory sticks that have been damaged or electronically erased. In *Proceedings of the 1st international conference on Forensic applications and techniques in telecommunications, information, and multimedia and workshop*, page 19. ICST (Institute for Computer Sciences, Social-Informatics and . . . , 2008.
 - [6] P. Rice and J. Moreland. Tunneling-stabilized magnetic force microscopy of bit tracks on a hard disk. *IEEE Transactions on magnetics*, 27(3):3452–3454, 1991.

5. PRIDOBIVANJE PODATKO IZ POŠKODOVANIH TRDIH DISKOV

Tudi podatki na trdem disku nam lahko povedo določene informacije o osumljencu. Naloga forenzičnega preiskovalca je skrbno preučiti podatke in natančno združiti zgodbo. Dato-teke na trdem disku in metapodatki, ki jih definirajo, nam lahko razkrijejo, kako in kdaj je bil disk uporabljen. Večina trdih diskov s katerimi se srečujejo forenzični preiskovalci je popolnoma zdrava. Vendar pa obstajajo situacije, ko je nekdo namerno poskušal uničiti podatke na trdem disku. V primeru da gre za fizično poškodbo diska obstajajo tehnike mikroskopiranje magnetnih sil s katerimi lahko rekonstrui-ramo podatke, ki so bili zapisani na disk [6].

6. ZAKLJUČEK

V članku je bilo analiziranih več vidikov poškodb naprav. Identificirani so bile dobre prakse ravnanja z napravami, kadar so te podvržene določenim tipom poškodb. Kljub temu, da so naprave poškodovane je iz njih v veliki večini primerov še vedno mogoče pridobiti podatke, ki nam lahko v okviru forenzične preiskave pomagajo pri iskanju določenih odgovorov. Opisali smo načine pridobitve podatkov iz pametnih telefonov oziroma mobilnih naprav, pri tem pa moramo upoštevati stopnjo poškodovanosti naprave. Prav tako je v določeni meri mogoče pridobiti podatke iz poškodovanih flash pomnilnikov ter trdih diskov, vsekakor pa je stopnja učinkovitosti pridobitve podatkov odvisna od poškodovanosti naprave.

7. VIRI IN LITERATURA

- [1] Damaged device forensics. https://www.dfrws.org/sites/default/files/session-files/pres_damaged_device_forensics.pdf.pdf. Dostopno: 18.4.2019.

Image tamper detection based on demosaicing artifacts

Matjaž Mav

Faculty of Computer and Information Science,
University of Ljubljana
mm3058@student.uni-lj.si

Klemen Jesenovec

Faculty of Computer and Information Science,
University of Ljubljana
kj9807@student.uni-lj.si

ABSTRACT

This paper reviews and summarizes research done on image tampering detection using a color filter array demosaicing technique. The algorithm and its features are explained and implemented in MATLAB. The algorithm is tested on sample data and results are discussed.

Keywords

Image tampering, digital forensics, color filter array

1. INTRODUCTION

This paper aims to review and summarize a paper about image tamper detection in the field of digital image forensics [2] (later referred to as referenced paper).

The paper classifies image tampering into three groups:

Targeted Tamper Detection Which focus on detecting specific groups of tampering such as re-compression, cloning, resizing, etc.

Universal Tamper Detection Which can only determine if the image was tampered with or not.

Localized Tamper Detection Which can identify parts of the image that were tampered with.

The paper describes a Color Filter Array (CFA) demosaicing based tamper detection that can detect both local and universal tampering. The method works due to the fact that image tampering usually alters CFA demosaicing artifacts in a way that can be detected and can thus detect a wide range of tampering techniques.

In the following sections we explain CFA, summarize the technique presented in the referenced paper and present our own implementation in MATLAB. Finally, we discuss the results and limitations of our implementation and compare it to the results in the referenced paper.

2. COLOR FILTER ARRAY

Color filter array is a grid of small color filters put over an image sensor. The filters are needed due to the sensor's inability to capture wavelength information and thus cannot distinguish between different colors. The CFA filters out specific colors before the light arrives at the image sensor. This enables cameras to capture color information.

There are many types of CFA although the Bayer filter is most commonly used and is illustrated in figure 1. It uses one red, one blue and 2 green filters per pixel. Green being used twice due to human eye sensitivity to green color.

Other types of filters include a RGBE filter which also uses an emerald filter, a CYYM filter which uses one cyan, two yellow and one magenta filter, a RGBW filter which also uses a white filter, etc.

The process of interpolating different sets of color information into a single image is called demosaicing.

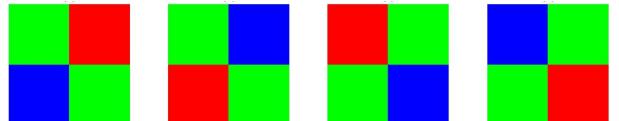


Figure 1: Commonly used Bayer filters

CFA interpolation or demosaicing [4, 3] is the process of building a single image from multiple color sets. This can be achieved using simple interpolation techniques such as nearest-neighbor interpolation, bilinear interpolation, bicubic interpolation, spline interpolation, etc.

Advanced demosaicing algorithms using pixel correlation such as variable number of gradients, pixel grouping, adaptive homogeneity-directed are out of scope of this paper.

Bilinear interpolation is used in this paper as it was also used in the referenced paper. The referenced paper also states result do not differ using bicubic interpolation.

In our MATLAB implementation we used interpolation implementation by S. Leszek.[4].

3. CFA BASED FEATURES

In this section we will review two CFA based features introduced in the paper [2]. The first feature (referred to as F_1) mainly relies on the estimation of the source digital camera CFA pattern. The second feature (referred to as F_2) relies on the source digital camera sensor noise.

Both of the features can be used in universal tampering detection and also when applied with sliding window in local tampering detection.

For both of the features we implemented algorithm in MATLAB. All code is available on the GitHub repository [matjazmav/fri-1819-df-seminar](https://github.com/matjazmav/fri-1819-df-seminar)¹.

3.1 Feature 1: Pattern number estimation

This feature tries to estimate which CFA pattern was used by the source digital camera.

To estimate possible used CFA pattern, the image is first re-sampled with each of the CFA patterns (usually only 4 most commonly used that are illustrated in figure 1) and then re-interpolated with the same CFA pattern. This steps are illustrated in figure 2. After that (3) MSE between the image and its re-interpolated part is calculated. For the image that is not tampered, one of the MSE values are expected to be significantly smaller from the others. If that is not the case, the image may have been post-processed.

We followed steps that are introduced in referenced paper and implemented MATLAB algorithm that estimates F_1 feature. Unfortunately we were not able to reproduce the results that are described in there.

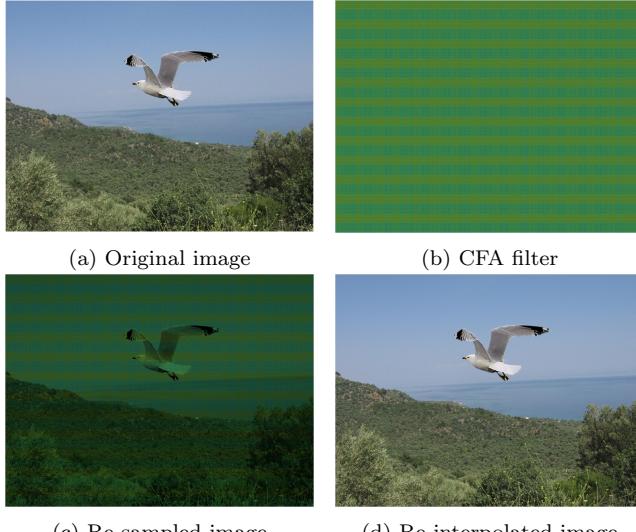


Figure 2: Steps taken in F_1 feature estimation. Original image (a) and selected CFA filter (b) are used to re-sample image. The re-sampled image (c) is then re-interpolated with the selected CFA filter (b). Final result is shown in (d).

3.2 Feature 2: CFA based noise analysis

This feature analyses the source digital camera noise. If the image is CFA interpolated the variance of the CFA interpolated pixels is expected to be significantly smaller than the variance of non-interpolated pixels.

To calculate F_2 feature we need to follow this steps:

1. Estimate sensor noise on the green channel of the image. First denoise image with level 1 wavelet denoising. Noise is defined as difference between image and corresponding denoised image.
2. Calculate variance of the estimated noise of the interpolated A_1 and corresponding non-interpolated A_2 pixels.
3. The final F_2 feature is calculated as maximum between both variance ratios (equation 1).

$$F_2 = \max\left(\frac{\text{var}(A_1)}{\text{var}(A_2)}, \frac{\text{var}(A_2)}{\text{var}(A_1)}\right) \quad (1)$$

Our MATLAB implementation of the F_2 feature extraction for the global tampering detection follows as:

```

1 function f2 = extract_f2(I)
2 % INPUTS
3 % I - RGB image
4 % OUTPUTS
5 % f2 - f2 measure
6
7 [h,w,~] = size(I);
8
9 G = double(I(:,:,2)); % extracted green color channel
10 Gd = wdenoise2(G,1); % denoise image
11 N = G - Gd; % extract image noise
12
13 % resize green color mask
14 mask = logical(repmat([1 0; 0 1], ceil([h/2 w/2])));
15 mask = mask(1:h, 1:w);
16
17 A1 = N .* mask; % extract non-interpolated noise vector
18 A2 = N .* ~mask; % extract interpolated noise vector
19
20 % calculate f2 measure
21 vA1 = var(A1(:));
22 vA2 = var(A2(:));
23 f2 = max(vA1/vA2, vA2/vA1);
24
25 end;

```

With slight modification of upper implementation, lines 13 - 23 were processed with sliding window (using function `blockproc`), we were able to calculate F_2 map and successfully localize tampering on the two images that were used in referenced paper.

4. EXPERIMENTS

We evaluated both of the features on dataset used in referenced paper [2], copy-move dataset MICC-F220 dataset [1] that contains 110 pristine and 110 tampered images and also on small dataset that we prepared².

Because of mostly unsuccessful results we also evaluated our implementations on the UCID dataset [5]. UCID dataset contain uncompressed color images in TIFF image format, taken by Minolta Dimage 5 digital colour camera (uses G -

² Available on the GitHub repository in folder `resources/datasets/ours`.

¹<https://github.com/matjazmav/fri-1819-df-seminar>

R - G - B CFA). Since referenced paper mentions that this two purposed techniques are strongly sensitive on used CFA, JPEG compression and resizing, we tried to tamper raw images with copy-move technique without resizing and using different compression levels saving into JPEG and PNG image format. We were able to produce a tampered image that could be detected with our implementation.

4.1 Local tampering detection

As we described in section 3.2, we were able to successfully replicate the local tampering detection on the images used in the referenced paper. The results of the our implementation are illustrated on the figure 3. Since our results are very similar to the results illustrated in the referenced paper we assume that our implementation is correct. Result on the image we produced using copy-move tampering is displayed on figure 4.

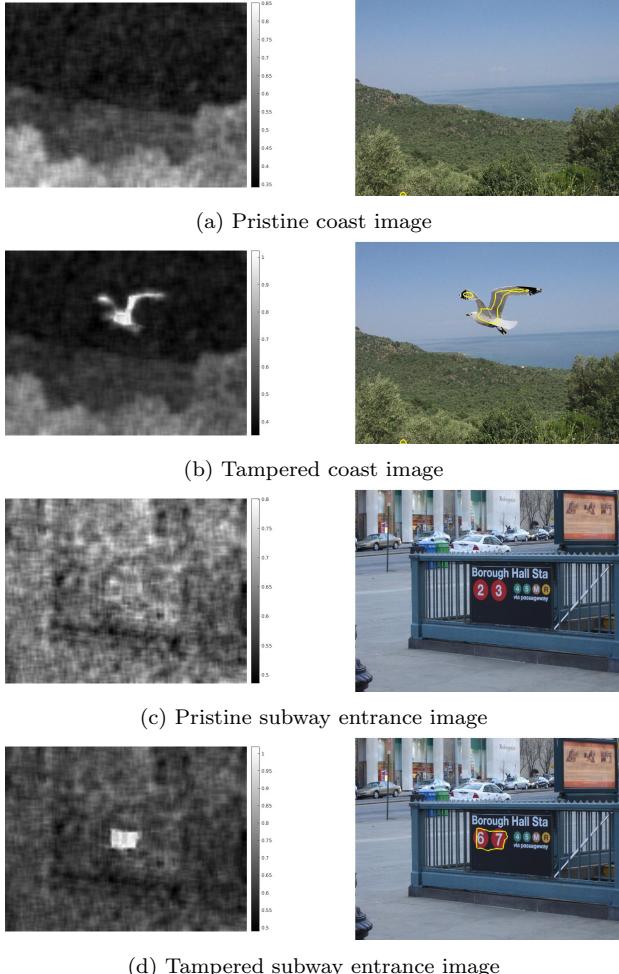


Figure 3: Local tampering detection with F_2 feature. Left images visualize $1/F_2$ map and right images are annotated where map is greater than threshold $T = 0.8$.

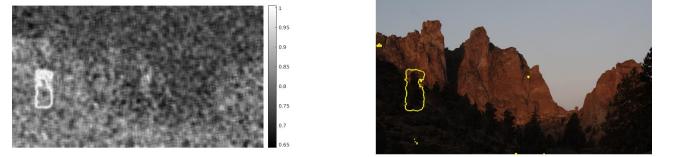


Figure 4: Local tampering detection with F_2 feature on the image we produced using a copy-move modification.

We discovered that this method is highly sensitive to image compression. The change of results due to compression is illustrated on figure 5.

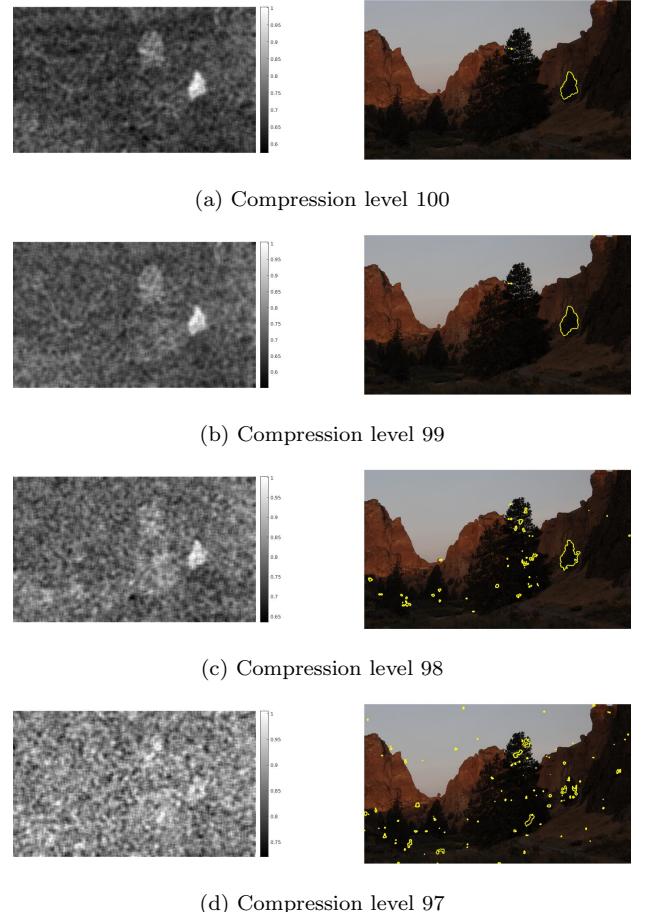
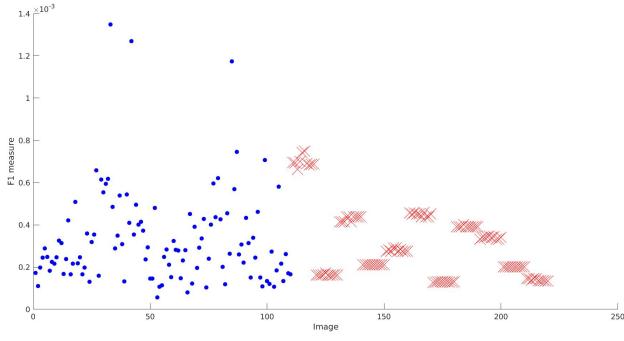


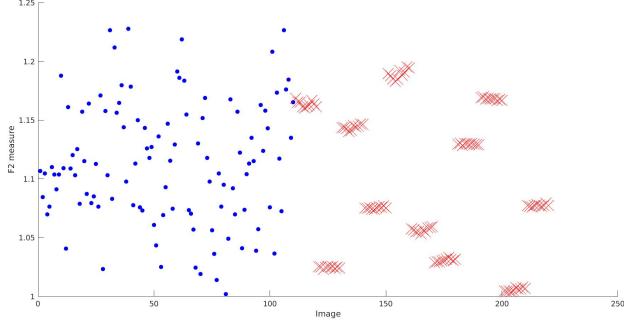
Figure 5: Effects of compression on the F_2 feature. Algorithm works flawlessly on the picture without compression but breaks completely with only 97% compression.

4.2 Universal tampering detection

Our implementation is not able to classify between pristine and tampered images. We tested both of the features on MICC-F220 dataset and illustrated extracted feature values in figure 6.



(a) Feature 1



(b) Feature 2

Figure 6: Universal tampering detection with F_1 and F_2 features on MICC-F220 dataset.

● pristine image
 ✕ tampered image (copy-move)

5. CONCLUSIONS

As seen in the results we were able to reproduce the results achieved in the referenced paper and on the image we produced. However we were not able to detect tampering on other datasets. This could be a result of improper post-processing done on the tested datasets such as: (1) use of an improper tampering technique (we mostly used copy-move, some of examples also included edge blurring and resizing), (2) use of too advanced image editing tool (we used Gimp and Adobe Photoshop), or (3) use of an improper compression levels and image formats as shown above.

6. REFERENCES

- [1] I. Amerini, L. Ballan, R. Caldelli, A. Del Bimbo, and G. Serra. A sift-based forensic method for copy-move attack detection and transformation recovery. *IEEE transactions on information forensics and security*, 6(3):1099–1110, 2011.
- [2] A. E. Dirik and N. Memon. Image tamper detection based on demosaicing artifacts. In *2009 16th IEEE International Conference on Image Processing (ICIP)*, pages 1497–1500. IEEE, 2009.
- [3] B. K. Gunturk, J. Glotzbach, Y. Altunbasak, R. W. Schafer, and R. M. Mersereau. Demosaicing: color filter array interpolation. *IEEE Signal processing magazine*, 22(1):44–54, 2005.
- [4] S. Leszek. Cfa interpolation detection. 2009.
- [5] G. Schaefer and M. Stich. Ucid: An uncompressed color image database. In *Storage and Retrieval Methods and Applications for Multimedia 2004*, volume 5307, pages 472–481. International Society for Optics and



5 METODE METHODS

Using deep learning approach for malware classification

Jan Gašperlin
jg1724@student.uni-lj.si

Blaž Beličić
bb7117@student.uni-lj.si

Jakob Makovac
jm6531@student.uni-lj.si

ABSTRACT

In this paper we present work done by researchers regarding malware classification using deep neural networks. Their approach used CNN in combination with LSTM neural network to statically analyze malware binary code, which was beforehand transformed into gray-scale picture. Their work improves upon existing techniques and drastically decreases entry level for non expert personnel, which can effectively use proposed deep neural network model. It achieved final accuracy score of 98.8 % on validation data.

Keywords

Machine learning, Malware, Deep Neural Networks, Long-Short Term Memory

1. INTRODUCTION

This paper describes work done in field of malware classification using machine learning. It will mostly cover work done in paper "Deep learning at the shallow end" [6] with some additional input.

Machine learning methods as deep neural networks are well researched and used in multiple different domains. This includes, but not limited to, weather, medicine, transportation, news and finances. Today we can confidently say that soon all domains will use some form of machine learning in the future to increase efficiency.

This is crucial since as mentioned in paper law enforcement agencies around the world are facing problems with ever increasing amount of digital evidence and lack of experts capable of analyzing large amount of data. With growing software complexity amount of malware is rising. The security vulnerabilities database is only getting larger at increasing speed. This is why there is a need for easily accessible forensic tools with low barrier of entry. Most related works described in paper need an expert, contrary to their proposed approach, where only training of classifier needs expert knowledge while usage has low barrier of entry. Although implementing neural networks represents difficult

challenge and needs extensive domain knowledge to obtain optimal performance. This represents even bigger challenge when trying to build classifier for malware, where input is machine code. Input therefore has dynamic length which needs to be converted to fixed length. With this method we always lose some data, that could be crucial for making predictions.

Our paper will explain how researchers, explored related works, implemented their own solution and evaluated their classification model.

2. RELATED WORKS

As we already mentioned need for automatic analysis of digital evidence is growing every day. This is mostly evident in evidence backlog. Processing of evidence can take long enough that age of suspect comes in question. There are cases that need to be dismissed when uncertainty arises due to person ageing.

To alleviate some of these problems there were attempts to automate at least part of digital forensic process. In this paper they mention attempts that use some form of machine learning to reduce manual work and increase detection accuracy. To train such models in our case neural network, others mostly used data-sets that contain labeled malware. For binary classification labels used were malicious or benign, while for multi-class classification the malware types were used.

Most presented approaches using deep neural networks required expert knowledge to extract required features from malware. This was done by analyzing raw machine code and executing it in sandbox environment, where behaviour was analyzed. This approach can take significant amount of time, which was reason why researchers decided to use only machine code as input.

One of mentioned approach attempted to use malware API calls [5] as features extracted from binary code and then used in training neural network. Some work also went into extracting more refined features from disassembled code, since they also included libraries. All of the above methods use heavy prepossessing steps that also require domain expert knowledge.

There was also case when a research team [7] used only machine code, but they designed deep learning architecture that was able to handle long input sequences. While on the

other hand mentioned solution in paper used fixed input size, which they achieved using image resizing techniques. This way they were able to retain enough details for neural networks to still effectively learn to classify malware.

One other attempt [3] used convolutions neural network and image down sampling to 32x32 picture. Key difference between this approach and the one used in paper was use of LSTM (Long-Short Term Memory) on top of CNN (Convolutional Neural Network) and larger input picture. Which resulted in better classification accuracy.

3. METHODOLOGY

In this section we describe the approach the researchers used to tackle the problem of malware classification. We describe the dataset that was used, data preprocessing, neural network architecture and the design of the experiment.

3.1 Dataset

Dataset used for this research was a part of Microsoft Malware Classification Challenge.[8] The competition itself finished in 2015, the dataset however is still available on Kaggle. It features 10868 samples of known examples of malware software grouped into 9 families. Each data point features the hexadecimal representation of file's binary content. Header is not included in the binary content. Each data point also includes metadata containing function calls, strings and similar, generated by IDA disassembler. Researchers chose a deep learning approach and aimed for minimal expertise required for feature construction. Consequently they used only raw binary content for classification.

3.2 Preprocessing

The choice of classification method dictated the preprocessing procedure that was required before data could be used for training the model. Deep learning approach provides a couple of advantages. One of them is a short classification time after the model is trained, although this is a double-edged issue, because training can take quite a long time. The other advantage is that feature construction does not require domain knowledge. The input for deep neural network can be raw data.

There are however also a couple of drawbacks when using a deep learning approach. The first is already mentioned very computationally heavy training of the model. The other is more relevant to preprocessing procedure. Deep neural networks require all input vectors to be of the same length. The length of compiled malware is unfortunately highly variable as can be seen in Figure 1.

There are many available methods for unifying input vectors to be the same size. In this research they used a generic image scaling algorithm, where byte stream was interpreted as a one dimensional image. Despite the fact this is a lossy compression of data, it preserves spatial patterns that might be present in the data. The advantage of one dimensional image representation over 2D image representation is the preservation of information order. Researchers decided to represent every input vector as 10000 byte values.

3.3 Architecture

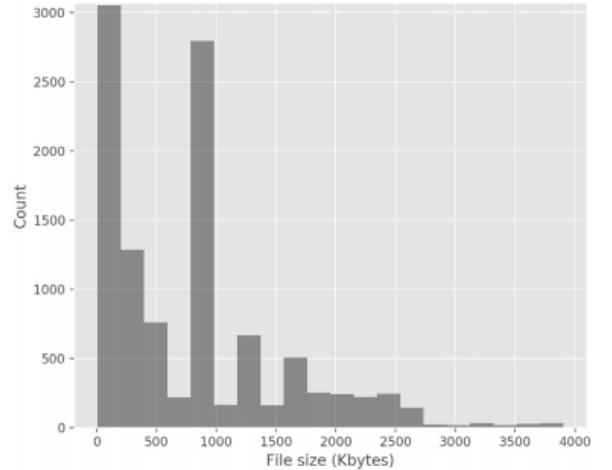


Figure 1: Distribution of file size in kilobytes of the raw binary content of files.

When utilizing a deep learning approach it is of crucial importance to choose a correct neural network architecture to ensure the best possible results. The choice largely depends on the nature of the data. Researchers chose three different architectures for their models.

All three models start with multiple convolutional layers to extract all the spatial features of the data. After that, the first model consist of a softmax layer that classifies the input into one of the classes as seen in Figure 2.

The other two models try to extract additional information out of input data. We are trying to classify computer code, so we are expecting to find dependencies between different parts of the data stream. Recurrent neural networks are very good at predicting such data, so the other two models make use of that fact. They both employ Long Short Term Memory (LSTM) modules between convolutional layers and output layer. The difference is that one uses one LSTM layer that models forward dependencies and the other uses two LSTM layers, one in each direction so it can also capture backward dependencies as seen in Figures 3 and 4. LSTM modules are different from feed-forward neural networks because they feature feedback connections. This gives them the ability to process sequences of data unlike feed-forward networks that can only process single data points.

3.4 Experiment Design

After the preprocessing step researchers had 10860 labeled samples. They felt that number is on the low side, so they opted for a more robust method and used five fold cross-validation. That means the dataset was split into five parts. The model was then trained on four of those parts and validated on the part that was left out of training. It is important that the splitting is done randomly, so all the parts have a similar class distribution as the original dataset. The distribution of classes in the dataset was highly imbalanced, so the researchers took two steps to mitigate the effect of imbalance on the model accuracy. The first one was to use the averaged F1-score for each of the classes to estimate the performance of a given model. F1-score is the harmonic mean of precision and recall. This way they prevented the

Table 1

The number of parameters and the training time for the specified deep learning Configurations.

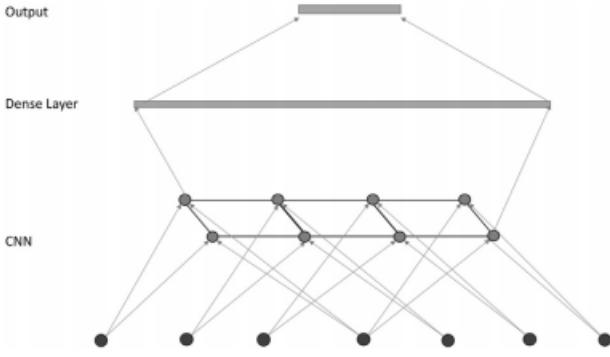


Figure 2: Convolutional Neural Network architecture.

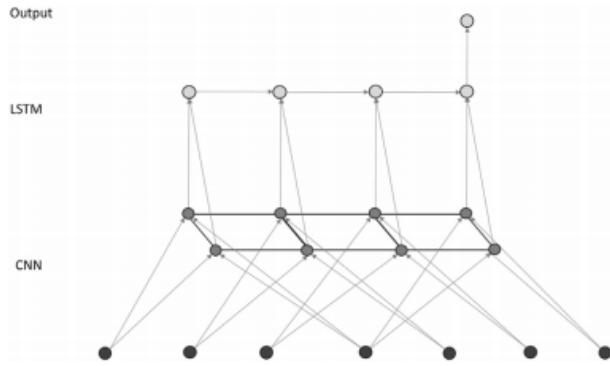


Figure 3: Convolutional Neural Network architecture with unidirectional LSTM layer.

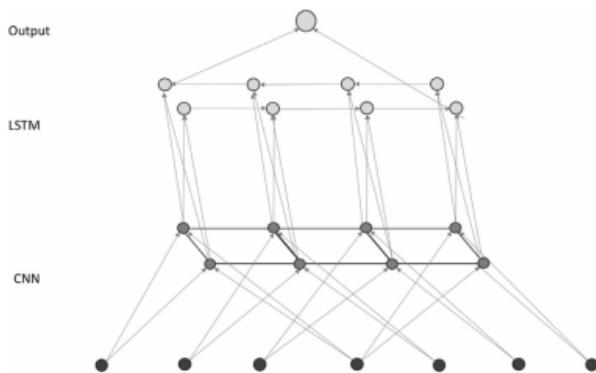


Figure 4: Convolutional Neural Network architecture with bidirectional LSTM layer.

Configurations	No Params	Train time (m)
CNN - Def Sampl	1,842,069	5.6
CNN - Reb Sampl	1,842,069	10.1
CNN UniLSTM - Def Sampl	155,669	32.1
CNN UniLSTM - Reb Sampl	155,669	55.1
CNN BiLSTM - Def Sampl	268,949	62.1
CNN BiLSTM - Reb Sampl	268,949	106.2

model from favoring the majority classes over the minority ones, since F1-score treats all classes with equal importance. The other step was to improve sampling which is again important to prevent model from training mainly on the examples of majority classes. They solved this with generating data batches for training. Each batch then drew roughly the same number of random examples from each class.

When using deep learning models, overfitting is a very common problem. To combat this, researchers chose L2 regularization to constrain weights on convolutional layers and dropout in the dense and LSTM layers, which promotes the model to make use of the largest number of neurons to represent knowledge during training. Hyperparameters were chosen during cross-validation and the final model was then trained on the best configuration and the whole dataset.

4. RESULTS & DISCUSSION

4.1 Results

All deep learning models have three convolutional layers with rectified linear unit activation function. There are 30 filters at first layer, 50 at second and 90 at third. The outputs of convolutional layers are connected to a layer of 256 units. For complete configuration each architecture will be paired with one of the data batch generators: the default sampling batch generator and the class rebalance batch generator. Models are implemented using the Keras library with the Tensorflow backend.

Each model is trained for 100 epochs on Nvidia 1080 Ti graphic card. Model weights are modified with Adam optimization method [4] to minimize the average logarithmic loss criteria (measure of the performance of a classification model).

In Table 1 we see number of parameters and training time for each of deep learning configuration. Further, in Table 2 we see the average accuracy and F1 score for different configurations. The best score achieves CNN BiLSTM with the class rebuilding sampling procedure, with 98.20% accuracy and 96.05% F1 score. It was chosen for final model and trained on the entire training data set. Choosing the best model among 100 epochs, 90% of data set is used for tuning model weights and remaining 10% for data validation.

Final convolutional neural network BiLSTM with the class rebuilding sampling procedure model, achieves an average log-loss of 0.0762 on the validation data and accuracy of 98.80%. After that testing was applied on data set with malware files from Kaggle. They receive a public score of 0.0655 and a private score of 0.0774 for average log-loss. It looks like that model generalizes well on new data.

In Table 3 we see the time taken for CNN BiLSTM to pre-

Table 2

Average accuracy and F1-score of different deep learning configurations using the 5-fold cross-validation procedure.

Deep Learning Conf	Acc (%)	F1 (%)
CNN - Def Samp1	95.1	92.14
CNN - Reb Samp1	95.8	92.14
CNN UniLSTM - Def Samp1	97.64	94.15
CNN UniLSTM - Reb Samp1	98.12	95.92
CNN BiLSTM - Def Samp1	97.91	95.52
CNN BiLSTM - Reb Samp1	98.20	96.05

Table 3

Time to pre-process and predict binary files in the test set with the final model.

Stages	Total (s)	Average (s)
Convert bin file to 1D rep	191.22	0.0176
Prediction	23.1	0.0021

process and predict malware classes for the 10,873 test files.

4.2 Discussion

One dimensional representation of the raw binary file is good for the malware classification, because it preserves the sequential order of the code. It is very similar to image representation of the malware binary file. However, it is simpler and does not need to decide about the image ratio.

Class re balance sampling procedure used in the article improve accuracy and F1 score for all the convolutional neural network long short term memory configurations (CNN LSTM). The reason could be, that samples from all classes in each batch are included. Back propagation then receives a better signal for tuning the models parameters.

Best performance is achieved when training the CNN BiLSTM with re balance sample. CNN model is better at utilizing GPU efficiently comparing to CNN BiLSTM. Also with the batch sampling procedure, training CNN is 10 times faster. CNN BiLSTM uses 268,000 parameters, on the other hand CNN uses 1.84 million. CNN BiLSTM model is only 1.5 times slower, when predicting the classes of raw binary files.

CNN UniLSTM model with class re balance is good compromise between both. Training is quicker compared to CNN BiLSTM and performance is also quite good. But performance of the deep learning model is better, when another dependency direction is added in the binary code. It is better to use both forward and backward layer (CNN BiLSTM model), than only forward layer (CNN UniLSTM model). Also bigger improvement is achieved with CNN LSTM architecture, instead of using only CNN.

Ahmadi et al. [1] were using machine learning approach for malware classification. They also tested it on files from Kaggle dataset. It was combination of different features extracted from the raw binary files and the disassembled files. They use 5-fold cross-validation procedure. One of the feature was image representation of the raw binary files, which is similar to the one dimensional representation of the raw binary file used in this article. Ahmadi et al.[1] model obtains 95.2% accuracy. That indicate that CNN approach is

slightly better.

Another advantage of deep learning approach is the time taken to classify a new binary file. Training models need a GPU, but once that is done, the final model needs only CPU, to predict the malware class of a new binary file. Final deep learning model takes on average 0.02 second for classification on 6 core i7-6850K Intel processor. This includes conversion and prediction. For comparison, machine learning model from Ahmadi et al. [1] needs on average between 0.75 and 1.5 second.

In another article which also uses similar convolutional neural network approach from Gibert et al. [2], they achieved public score of 0.1176 and private score of 0.1348. They were using image representation of raw binary file with 34.5 million parameters. CNN BiLSTM model used in this article achieved public score of 0.0655 and private score of 0.0774 with 268,000 parameters.

5. CONCLUSION

Deep learning approach presented in the article achieves performance of 98.2% in the cross-validation procedure and 98.8% accuracy tested on the validation data. Approach does not require feature engineering. This makes it simpler for researches, who are not field experts. Another good feature is a short classification time of binary file. It takes only 0.02 second, which makes it practically useful.

Results showed that we could use model in practice an improve it with new training data.

One dimensional representation of the raw binary has also some drawbacks. It does not consider the semantics of the binary code in the raw binary file. Experiment shows, that there are spatial patterns in each of malware class in the binary files. Deep learning models could use this patterns to class the malware.

5.1 Future work

For future work, deep learning approach should be tested on malware data sets with more malware classes. Also we could preserve the semantic meaning of each byte in the file in the preprocessing step. Another feature we could add, is modification of the model that could detect if the new binary file belongs to one of the available classes or belongs to a new malware class. In the future we could apply more complex deep learning architectures to achieve better accuracy.

6. REFERENCES

- [1] M. Ahmadi, D. Ulyanov, S. Semenov, M. Trofimov, and G. Giacinto. Novel feature extraction, selection and fusion for effective malware family classification. *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy*, pages 183 – 194, 2016.
- [2] D. Gibert. *Convolutional Neural Networks for Malware Classification*. Master’s thesis, Universitat Politècnica de Catalunya., 2016.
- [3] D. Gibert Llauradó. Convolutional neural networks for malware classification. Master’s thesis, Universitat Politècnica de Catalunya, 2016.
- [4] D. Kingma and J. B. and. Adam: A method for stochastic optimization. *3rd International Conference for Learning Representations*, 2015.

- [5] B. Kolosnjaji, A. Zarras, G. Webster, and C. Eckert. Deep learning for classification of malware system call sequences. In B. H. Kang and Q. Bai, editors, *AI 2016: Advances in Artificial Intelligence*, pages 137–149, Cham, 2016. Springer International Publishing.
- [6] Q. Le, O. Boydell, B. M. Namee, and M. Scanlon. Deep learning at the shallow end: Malware classification for non-domain experts. *Digital Investigation*, 26:S118 – S126, 2018.
- [7] E. Raff, J. Barker, J. Sylvester, R. Brandon, B. Catanzaro, and C. K. Nicholas. Malware detection by eating a whole exe. In *Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
- [8] R. Ronen, M. Radu, C. Feuerstein, E. Yom-Tov, and M. Ahmadi. Microsoft malware classification challenge. *CoRR*, abs/1802.10135, 2018.

OpenForensics: A digital forensics GPU pattern matching approach for the 21st century

DFRWS 2018 Europe - Proceedings of the Fifth Annual DFRWS Europe

David Lorente Llinares
Faculty of Computer and
Information Science
University of Ljubljana
dl8614@student.uni-lj.si

Pasqual Martí Gimeno
Faculty of Computer and
Information Science
University of Ljubljana
pm2724@student.uni-lj.si

Tomáš Fišer
Faculty of Computer and
Information Science
University of Ljubljana
tf1593@student.uni-lj.si

ABSTRACT

It happens that filesystems become corrupted. Sometimes, it may be done intentionally by a person that committed a crime. In such a case, the process called File carving is used to recover files. This process is really useful during a digital investigation. In the paper for which we are giving a survey, the authors provided a new approach for recovering files. They came with a framework called *OpenForensics* which contains a parallel algorithm that process pattern matching on GPU. Their approach is based on *Parallel Failureless Aho-Corasick* algorithm (PFAC) which search for multiple patterns simultaneously. They also implemented the OpenForensics framework and compared it to commonly used digital forensics tools (Foremost, Recover My Files). Results show that their framework outperforms both tools. Furthermore, their algorithm performs the pattern matching almost as fast as the read speed of the storage device is.

Keywords

Digital Forensics, Processing model, Pattern matching, Asynchronous processing, GPU, GPGPU

1. INTRODUCTION

The technological advances of recent years have led to an increase in storage capacity in the devices which, at the same time, led to the inefficiency of pattern matching techniques used by Digital Forensics (DF) tools in order to analyze forensic data.

Since the first commercial DF analytical tools, the research efforts concentrated on improving pattern matching algorithms and the use of parallel hardware; failing, however, in bettering pattern matching techniques and processing technologies because of its little research or commercial interest. The pattern matching technique is key in many DF analysis where *File carving*, analysis of raw disk image to reproduce files when the metadata is corrupted or nonexistent, is needed.

The paper we review ([2]) intends to provide effective techniques for a complete analysis of DF corpora with an approach that makes use of all available computer resources, which boosts the processing time. For this, the authors focus on file carving, enhancing it with asynchronous parallelization of CPU and GPU in the processing step and handling string searching with the *Parallel Failureless Aho-Corasick algorithm*.

Our work will include not only an exposition of the methodologies, tests, and results of the authors but also basic concepts or technologies, necessary for a better understanding of the original paper, that will be described and related to previous work by other researchers in the same field.

2. PRELIMINARIES

In this section, we will explain several notions that were mentioned in the paper and were considered as commonly known. Firstly, we will explain file carving. Then, we describe algorithms for pattern matching which is useful in file carving. Lastly, we cover GPU processing which improves computation time of parallel algorithms.

2.1 File Carving

When a user deletes a file, the entry in the filesystem metadata is removed only. It means that raw data of the file are still present on the disk. Even formatting or repartitioning of the disk might not affect raw data. Thus, one can still recover the deleted file using *File carving*, which is the process of recovering files if filesystem metadata are missing. This is done by processing raw data in clusters and identifying their meaning. One way is to search through file headers and check whether they match with any known file type (mp3, png, avi, etc.).

2.2 Pattern Matching

Pattern matching is a mechanism for checking and locating a specific sequence of tokens. This sequence of tokens is called a pattern and tokens could be characters of a string. In digital forensics, pattern matching is used for example for file carving where multiple file headers are considered as patterns that we search for. Algorithms that solve pattern matching problem on strings are called *String-matching algorithms*. In the paper, two such algorithms are mentioned, the Boyer-Moore algorithm and the Aho-Corasick algorithm.

2.2.1 Boyer-Moore Algorithm

The Boyer-Moore algorithm was developed by R. S. Boyer and J. S. Moore in 1977 ([3]). The algorithm applies a pre-processing on the pattern which is suitable in the case where the pattern is much shorter than the text or when the same pattern is searched in multiple texts. In the preprocessing phase, the algorithm searches common suffixes in the pattern and gathers them into a suffix table. Thus, the suffix table allows skipping unnecessary comparisons of characters which would not match anyway. In the searching phase, the algorithm starts by checking the last character of the pattern and continue comparisons to the left while there is a match. On the other hand, when the matching of characters fails, the pattern is shifted against the text by a value stored in the suffix table. The complexity of the searching phase is $O(mn)$. In case the pattern is not periodic, the algorithm does $3n$ comparisons in the worst case.

2.2.2 Aho-Corasick Algorithm

In 1975, A. V. Aho and M. J. Corasick invented the algorithm called the Aho-Corasick algorithm ([1]). Unlike the Boyer-Moore algorithm, the Aho-Corasick algorithm matches multiple patterns simultaneously. This is why this algorithm is widely used for dictionary searching. The algorithm works as follows. Firstly, the prefix table is computed for the patterns and stored into a trie. In this phase, the algorithm searches for the maximum-length suffix of the current substring that also is the prefix of some pattern in the trie. After the trie is generated, the algorithm process the text. The complexity of the algorithm is $O(n + l + z)$ where n is the length of text, l is the sum of the lengths of patterns and z is the number of output matches. This algorithm can be easily processed by multiple threads in parallel. For this purpose, the text is divided into smaller blocks which contain a sub-string that overlaps string in neighboring blocks. Then, each thread process distinct block of text while the generated trie is accessible for every thread.

2.3 GPU Processing

A graphics processing unit (GPU) is a core of graphics card which is present in computers or mobile devices to work with computer graphics more efficiently. GPU is better for this purpose than a central processing unit (CPU) because of its parallel structure. The parallel structure allows processing on large blocks of data in parallel which is also the perfect match for parallel algorithms. In the 21st century, scientist came up with General-purpose computing on GPU (GPGPU) which is a concept of using GPU to perform computations that are usually done by CPU. Mainly used platforms for GPGPU that allows direct access to GPU instructions and simplifies the development on GPU are CUDA¹ and OpenCL².

3. RELATED WORK

Recently, many researchers have published publications which are connected to the reviewed paper ([2]).

In [5] it is discussed how today's DF tools are becoming obsolete, as the authors of our reviewed paper mention, and

¹<https://www.nvidia.co.uk/object/cuda-parallel-computing-uk>

²<https://www.khronos.org/opencl/>

the need for the community to adopt standardized, modular approaches for data representation and forensic processing to avoid a future without reliable forensic analysis results. Similarly, in [9] the authors write about how the capacity of digital storage devices has been increasing at a rate that has left digital forensic services struggling to cope, causing current forensic tools to fail to keep up expected results.

Regarding the Aho-Corasick algorithm, it was improved in [7] and revisited by the same authors in [6]. The Parallel Failureless Aho-Corasick algorithm (PFAC) uses heuristics to achieve better efficiency. Authors of OpenForensics considered this algorithm as a base for searching file headers.

Discussing the topic of Parallel processing, in [11] the authors explore the algorithms and methods to use for development of file carving tools that will do their job much faster than the conventional DF tools, also considering the use of the GPU instead of (or working with) the CPU. In [8, 12], GPGPU approach was used to improve file carving. According to the authors of the paper, currently used digital forensics tools for file carving are Scalpel ([10]) and Foremost³.

In the paper [4] authors discussed about how important is for DF investigations the ability that the file carving process has to reassemble fragmented files and, after analyzing the process, they conclude that the most efficient approach is using candidate byte strings as objects for a multi-tier decision problem in which each object has to be quickly validated or discarded.

4. METHODOLOGY

This paper focuses on making fundamental changes to the three processing stages of file carving: file detection and processing method, data reading method, and file reproduction method.

4.1 File Detection and Processing Method

They propose a scalable asynchronous approach, employing processing threads that act independently of each other to queue, read, and process data. This approach is hypothesized to scale well with the processing power available on the analyst's computer, reducing the chances of bottlenecks by employing all available processing resources.

Thus, the main idea is to employ both GPU and CPU processing. When performing GPU processing, *Open Computing Language* is used. OpenCL is a framework for writing programs that execute across heterogeneous platforms consisting of several different processing units and hardware accelerators, providing a standard interface for parallel computing using both task and data-based parallelism. Thanks to the usage of this API, the searching can be done using all of the available resources: the GPU performs pattern matching whilst the CPU is tasked with validation and recording of files found.

Regarding multithreading, the available CPU threads are divided equally between the usable GPUs (e.g., on a system with 2 GPUs and an 8 logical core processor, each GPGPU

³<http://foremost.sourceforge.net/>

device is allocated 4 independent CPU processing threads). The level of threading employed, however, is limited by the amount of system memory available, each thread requiring around 150 MiB of RAM and VRAM to run.

4.1.1 File Detection and File Carving

The file detection method adopted by OpenForensics is two-staged. The first pass of the searching marks files found within data, storing patterns and locations in an array. These locations are then passed to the CPU, which scans the result array for a file header, and pairs it with a matching file footer, repeating this process until all segments have been scanned. Partial patterns are also accounted for by employing a windowed technique. The second pass optionally performs file carving of found files.

4.1.2 File Processing

OpenForensics employs the *Parallel Failureless Aho-Corasick* algorithm to accelerate GPU and CPU pattern matching operations. This algorithm, PFAC in short, comes from the Aho-Corasick algorithm, which is a dictionary-matching algorithm that locates elements of a finite set of strings within an input text, matching all strings simultaneously. This trait makes this algorithm especially useful when searching for large amounts of patterns within data as, again, all defined patterns are searched with a single read. Otherwise, other searching algorithms such as Boyer-Moore or KMP might be more suitable.

The Parallel Failureless version of the algorithm, however, is more complex and consists of two phases: the pre-processing of file headers and footers (patterns) to formulate a state transition table (STT), and the processing steps that both CPU and GPU devices followed.

The STT is generated before searching begins, and is processor agnostic, which means that both CPU and GPU implementations can follow it to look for their next instruction. For its generation, it requires an input of the patterns searched in the format of a byte array which, in the case of OpenForensics, is a 2D array of integers. In this array, rows are the *states* and columns represent *input values*.

The STT consists of four key state types, stored sequentially in the order hereby presented:

- Fail state: signals that no pattern matches have been found.
- Pattern state: provides a unique identifier for each pattern searched for.
- Initial state: used as an initial search state for each byte of data.
- Transition states: used to decide the next state based on the data value read. The number of transition states varies depending on the number and length of patterns searched

Algorithm 1 PFAC STT generation

```

1: patterns is a 2D array of byte
2: STT is a variable-length array of integer
3: row is an array[256] of integer
4: sort patterns array by byte length, a < b
5: add fail state row to STT (STT[0])
6: for i = 0 to num of patterns do
7:   add row to STT (pattern found state)
8: end for
9: add initial state row to STT
10: state  $\leftarrow$  num of patterns + 1
11: for i = 0 to num of patterns do
12:   walkindex  $\leftarrow$  num of patterns + 1
13:   if STT[walkindex][target[i][0]] = 0 then
14:     STT[walkindex][target[i][0]]  $\leftarrow$  state
15:     state += 1
16:   end if
17:   walkindex  $\leftarrow$  STT[walkindex][target[i][0]]
18:   for j = 0 to patterns[i] length do
19:     if STT length <= walkindex then
20:       add row to STT (transition state)
21:     end if
22:     if STT[i][patterns[i][j]] = 0 then
23:       if j != patterns[i] length - 1 then
24:         STT[i][patterns[i][j]]  $\leftarrow$  state
25:         state += 1
26:       else
27:         STT[i][patterns[i][j]]  $\leftarrow$  i + 1
28:       end if
29:     end if
30:     walkindex  $\leftarrow$  STT[i][patterns[i][j]]
31:   end for
32: end for
33: return STT

```

Figure 1: PFAC STT generation.

Each row of the STT has a column for the number of possible inputs the processor can read when searching. In the case of OpenForensics, 256 columns are created, that represent each possible value for a byte read in data.

		Input value							
		0	1	2	3	4	...	255	
Fail state	{}	0	0	0	0	0	0	0	
		0	0	0	0	0	0	0	
Pattern state	{}	0	0	0	0	0	0	0	
		0	0	0	0	0	0	0	
Initial state	{}	5	8	5	0	0	0	0	
		6	0	0	0	0	0	0	
Transition state	{}	1	0	0	3	0	0	0	
		9	0	0	0	0	0	0	
		0	0	0	2	0	0	0	

Figure 2: OpenForensics state transition table.

So, the algorithm begins on the initial state for each individual byte of data analyzed. The STT is then used by the processor to transition state depending on the value read. This process continues until the value read has a state value of 0 (which corresponds to the fail state) or until the state value is less than the initial state and no further transitions are available. This indicates a pattern match. When this happens, the starting byte is flagged in memory. We attached the pseudo-code (see Figure 1) for further understanding of how the STT is generated as well as an example of STT (see Figure 2). Then, the pseudo-code for processing steps of PFAC is in Figure 3).

It is important to remark that, even though OpenForensics uses the same approach to processing for both CPU and GPU, the latter is able to analyze significantly more bytes simultaneously, achieving greater processing efficiency.

Algorithm 2 PFAC processing steps

```

1: n  $\leftarrow$  segment length
2: initial state  $\leftarrow$  num of patterns + 1
3: for i = 0 to n do
4:   state  $\leftarrow$  initial state
5:   pos  $\leftarrow$  i
6:   while pos < n do
7:     state = STT[state, segment[pos]]
8:     if state = 0 then
9:       break
10:    end if
11:    if state < initial state then
12:      record position i as match for pattern state
13:    end if
14:   end while
15: end for
16: return found patterns

```

Figure 3: PFAC processing steps.

4.2 Data Reading Method

To take full advantage of storage devices, the authors of this paper carried out an investigation to discover the optimal settings to perform data reading in a DF context. Threads and queue depth were the largest influencing factors to performance. However, other variables were also considered, such as the size of the stream buffer and the amount of data read by each read instruction queued.

It was decided to allocate a single thread to read the data with a queue depth of 32 read instructions. Only a single thread was used so that it would not interfere with the performance of other asynchronous threads employed for processing. This decision may suppose a disadvantage depending on the system used but allowing a queue depth of 32 read instructions provided this single thread with enough queued instructions to make efficient use of the transfer-rate of the storage devices analyzed.

As for the other variables considered, stream buffer was an odd one. Both CPUs tested in their study possessed a fairly

large internal cache, so it was assumed that increasing the stream buffer size would have a positive effect on data transfer speed. However, experiments showed that this was not the case and that a size of 4 KiB was the most effective. Finally, a segment size of 1024 KiB was found to be optimal for each queued read task.

Once the best settings were known, they were trialled with several different storage devices, from traditional hard disk drives (HDDs) to solid-state drives (SDDs), and NVMe SSDs. The data transfer rates achieved ended up being similar to the sequential read performance stated by the storage device manufacturer.

4.3 File Reproduction Method

OpenForensics performs file reproduction by performing a second pass on the data analyzed, in which all the data between two points on the storage device that has been flagged to contain a file is extracted. This pass does not read all data from the storage device to memory to perform file carving, but only the data between matched file headers and footers. All this data is then saved to new files to be used by the investigator.

As for the way the files are read, the methodology is similar to the one employed in the first pass. An asynchronous thread is employed for each logical core on the CPU. For each file, an available thread is tasked to read the data between header and footer locations.

5. EVALUATION

To compare the performance of the OpenForensics searching methodology, the paper presents performance results of performing pattern matching on connected hard disk drives and a forensic image file. These results have been then compared with other currently available open-source and commercial file carving software: *Foremost* and *Recover My Files*, in performing the same tests.

Two test platforms were chosen: a high-end workstation PC (test platform A) and a high-end laptop (test platform B), whose specifications can be seen in Figure 4.

Measurements were taken on the time required to search for predetermined patterns for both *Foremost* and OpenForensics. *Recover My Files*, however, cannot be explicitly instructed to search for specific patterns, so a different experiment was designed for this software to achieve a reasonable comparison.

Two types of test were performed. The first test type was a raw forensic file analysis, where a *dd* image of an external storage device is analyzed from the slowest storage device for each test platform (called storage device 1). The second test performs pattern matching on storage device 2, the faster one, of each platform, where the data is physically accessed from the storage device to memory.

5.1 Analysis of a Raw Forensic dd File

The first test analyzes a 20 GiB forensic image file on storage device 1 of each platform.

Test Platform	A	B
Type	Desktop	Laptop
Operating System	Windows 10 Pro & Ubuntu GNOME 17.04	Windows 10 Pro & Ubuntu GNOME 17.04
Processor	Intel Core i7-5820K	Intel Core i7-7700HQ
Processor Specification	6 Core/12 Thread @ 3.8 GHz	4 Core/8 Thread @ 3.8 GHz
Memory	16 GiB DDR4 2400 MHz	32 GiB DDR4 2400 MHz
GPU	Nvidia Titan XP (12 GiB GDDR5X)	Nvidia 1070 (8 GiB GDDR5)
GPU Specification	3584 Cuda cores @ 1417 MHz	2048 Cuda cores @ 1442 MHz
Storage Device 1	2x Samsung Evo 940 250 GB SATA3 SSD (RAID0)	SanDisk Ultra II 960 GB SATA3 SSD
Storage Device 2	Samsung Evo Pro 256 GB NVMe PCIe SSD	Toshiba THNSN5256GPUK NVMe 256 GB SSD

Figure 4: Test platform specifications

OpenForensics finished analysis with the fastest times out the three programs, with no variation in results between repetitions. The time to complete analysis on test platform B was around double than the one on A, which collates to the theoretical read speed limits that the storage devices can read data. This indicates that the laptop (platform B) may have been able to analyze much faster storage devices with the employed pattern matching approach. The result of this experiment measured on platform A is depicted in Figure 5

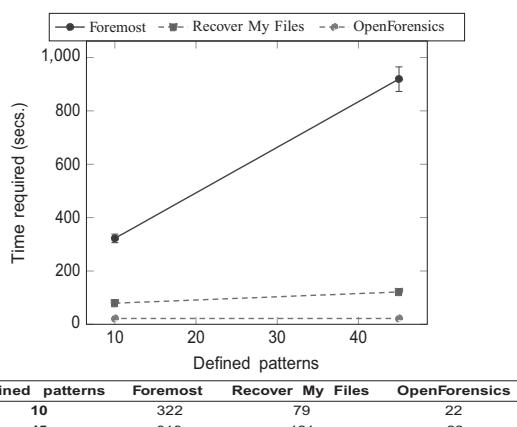


Figure 5: Test platform A: Mean time taken to conduct pattern matching on raw forensic dd file (secs.) with 95% confidence intervals.

5.2 Analysis of a Physical Storage Device

The second trial consisted in searching an entire physical drive.

Although both Foremost and OpenForensics were able to target the full physical drive, Recover My Files was only able to target visible partitions. In the end, OpenForensics manages to perform analysis the quickest and shows no variation in time when searching for 10 or 45 patterns. Results for both storage devices are similar in results. In this experiment, we show result for platform B only (see Figure 6).

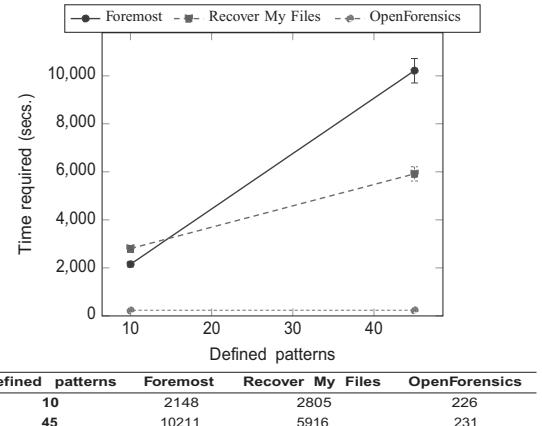


Figure 6: Test platform B: Mean time taken to conduct pattern matching on physical storage device (secs.) with 95% confidence intervals.

5.3 Processing Speed Analysis

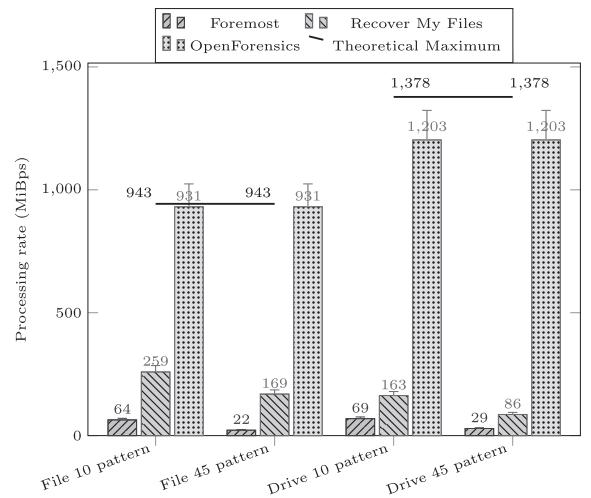


Figure 7: Test platform A file carver rate of analysis with 90% confidence intervals.

This section of their paper aims to analyze the speed of the pattern matching to establish whether the fastest method (that is, OpenForensics) achieves the maximum possible throughput from the storage devices tested. They assume that the maximum recorded sequential read speed of the storage device is the maximum rate in which analysis can be done. The result for test platform A is depicted in Figure 7.

In all test cases, OpenForensics is able to achieve the greatest rate of analysis, achieving a processing rate close to the sequential read performance of most storage devices tested. Specifically, it used 98.7% of the measured storage device sequential read performance at best, and 85.4% on average across all tests from the two test platforms. These results are, at the very least, four times better than the ones achieved with the other programs.

As a final note, they hypothesized that storage device transfer speeds would limit the possible rate of analysis when analyzing the forensic image from storage device 1 of each platform, and that analyzing the storage device 2 of each test platform would highlight limitations of current string searching approaches adopted in DF. Both hypotheses were evidenced to be true.

6. CONCLUSIONS

In this paper we explained key concepts about pattern matching techniques and data processing, relating them to previous work of other researchers on this field.

Furthermore, we presented the methodology used by the author's open source solution OpenForensics that improves pattern analysis on modern consumer hardware. We also show how the authors evaluated their software, comparing it to other commercial software like Foremost and Recover My Files that are, as can be seen in the results, outperformed by OpenForensics.

Finally, we should mention that the aim of the authors is that the methods they show in their paper serve as a framework for the next generation of DF tools accessible to professionals internationally.

7. REFERENCES

- [1] A. V. Aho and M. J. Corasick. Efficient string matching: An aid to bibliographic search. *Commun. ACM*, 18(6):333–340, June 1975.
- [2] E. Bayne, R. Ferguson, and A. Sampson. Openforensics: A digital forensics gpu pattern matching approach for the 21st century. *Digital Investigation*, 24:S29 – S37, 2018.
- [3] R. S. Boyer and J. S. Moore. A fast string searching algorithm. *Commun. ACM*, 20(10):762–772, Oct. 1977.
- [4] S. L. Garfinkel. Carving contiguous and fragmented files with fast object validation. *digital investigation*, 4:2–12, 2007.
- [5] S. L. Garfinkel. Digital forensics research: The next 10 years. *digital investigation*, 7:S64–S73, 2010.
- [6] C. Lin, C. Liu, L. Chien, and S. Chang. Accelerating pattern matching using a novel parallel algorithm on gpus. *IEEE Transactions on Computers*, 62(10):1906–1916, Oct 2013.

- [7] C. Lin, S. Tsai, C. Liu, S. Chang, and J. Shyu. Accelerating string matching using multi-threaded algorithm on gpu. In *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*, pages 1–5, Dec 2010.
- [8] L. Marziale, G. G. Richard, and V. Roussev. Massive threading: Using gpus to increase the performance of digital forensics tools. *Digital Investigation*, 4:73 – 81, 2007.
- [9] P. Penrose, W. J. Buchanan, and R. Macfarlane. Fast contraband detection in large capacity disk drives. *Digital Investigation*, 12:S22–S29, 2015.
- [10] G. G. Richard III and V. Roussev. Scalpel: A frugal, high performance file carver. In *Proceedings of the 2005 Digital Forensics Research Workshop*, 01 2005.
- [11] N. Škrbina and T. Stojanovski. Using parallel processing for file carving. *arXiv preprint arXiv:1205.0103*, 2012.
- [12] X. Zha and S. Sahni. Fast in-place file carving for digital forensics. In X. Lai, D. Gu, B. Jin, Y. Wang, and H. Li, editors, *Forensics in Telecommunications, Information, and Multimedia*, pages 141–158, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.



6 OMREŽNA IN MOBILNA FORENZIKA NETWORK AND MOBILE FORENSICS

Forensic Analysis of Mobile Applications on Android Devices with Fordroid

Digital Forensics 2018/19

Vanda Antolović

Faculty of Computer and
Information Science
University of Ljubljana
Ljubljana, Slovenia

va2351@student.uni-lj.si

Gojko Hajduković

Faculty of Computer and
Information Science
University of Ljubljana
Ljubljana, Slovenia

gh8590@student.uni-lj.si

Olivera Perunkovska

Faculty of Computer and
Information Science
University of Ljubljana
Ljubljana, Slovenia

op7933@student.uni-lj.si

ABSTRACT

Over the course of the past two decades, phones stopped just being phones. Nowadays, phones are not only used to send voice and text messages but also to browse Internet [2] and send emails, take photos, listen to music, get directions to anywhere you need to go, play video games, pay groceries (and bills), and more. Mobile devices have become an essential part of our lives, but also have become a part in criminal activities. As the involvement in these criminal activities increases, the need to rapidly tackle these activities increases. Digital forensics can be defined as the process of collecting, examining, analyzing and reporting of digital evidence without any damage. Digital forensics requires a detailed examination of devices such as computers, mobile phones, sim cards, tablets that contain digital evidence. However, traditional forensic approaches, which are based on the manual investigation are not scalable to numerous mobile applications. So this paper is presenting a *fully automated* tool named **Fordroid** for the forensic analysis of mobile applications on Android. This tool conducts inter-component statistic analysis on Android APK's and builds control flow and data dependency graphs. Also identifies what and where the information is written in local storage, and how the information is stored by parsing SQL commands. **Fordroid** is tested on 100 randomly selected Android applications and from the analyses, it can be seen that the success rate on locating where the information was written is 98% for paths and it is able to identify the structure of all database tables.

Keywords

Digital forensics, static analysis, Android, Fordroid, local storage

1. INTRODUCTION

In today's time, mobile phones are something crucial to everyday activities, whether those are work-related, study re-

lated or leisure related, it has become that one without a mobile phone will be left out of the loop of activities. Hence, with all these usages, a mobile application processes numerous information on a daily basis, which is in big amounts stored locally on individual's phone [3]. All of this information can be collected without user's knowledge and used in malicious intents, but they can also be used to expose persons that participate in some criminal activities. With Android being the most used platform among the smartphones, it has become a necessity to develop applications that can detect leaks of this kind of information, but as well as to determine what is this sensitive data, location of where this data is stored as well as how this data is stored in local storage. These three are the main focus of the application presented in the paper we are reviewing here, the application is called **Fordroid**.

Fordroid application takes a static analysis approach to provide answers to these questions. The static analysis enables us to identify all possible behaviors of the application that is being analyzed with no human intervention needed. Besides static analysis, what is also very often used are dynamic analysis and manual reverse engineering. Dynamic analysis is running an application in a simulation environment, or on a test phone. The purpose is trying to manually reproduce all possible program paths over a period of time, but the main problem is some specific behaviors that are needed might not be discovered this way. The main difference between static and dynamic analysis is that static analysis provides a full picture off application's possible malicious behavior in contrast to just preventing those that manifest in the testing environment, this being exactly the reason why static analysis is the widely used approach. In comparison the manual reverse engineering approach is parsing the overall documents, or code, to determine how the application works, and where it stores the data, which is in the end a very time-consuming and tedious task.

The remainder of this paper is organized as follows, in section 2 is shown the Android app architecture. In section 3 the related work is mentioned, that was part of the motivation of the paper we are reviewing. Section 4 presents the Fordroid application, its architecture, and section 5 giving an overview of the evaluation given in the paper. With section 6 being the overall conclusion.

2. ANDROID APPLICATION ARCHITECTURE

In order for a better understanding of how and where security issues may occur on the Android system, this section gives a brief overview of the Android application's architecture. Android is based on Linux which provides usage of variety predefined security functionalities and easier creation of drivers for different types of devices. Each Android application is a different user in Linux based multi-user system and each one runs on a different process. This allows each process to have its own Virtual Machine hence an application's code runs in isolation from other applications. For our subject of interest especially important are essential building blocks of an application, Components, which are entry points through which user or system can enter the application[3]. Four types of components are present in application architecture, and those are:

- **Activity** - Each activity represents a screen with user interface, and is an entry-point for user-application interaction.
- **Service** - Purpose of services are to perform long-running operations in the background, i.e playing music in the application while user is in a different application.
- **Broadcast receivers** - Main purpose of broadcast receivers is to allow registration for applications or system events.
- **Content providers** - Sharing data from one application to another is done by Content providers which is presented as a middle-man layer between data and application layers.

This kind of Android system architecture presents a challenge in applying static analysis. Many questions regarding the architecture of the system arise while conducting static analysis, such as how to successfully track and capture all paths of control flow changes in the system, since Android is an event-based system whose control is driven by events in the environment[3]. Content providers allow sharing data among applications. Another feature of Android is that in an application, a component can start: a native system component, component which is from within the same application or a component that is from another application. This declares Android as the component based system that uses Inter-Component Communication (ICC) extensively. Capturing all changes in control and data flow along ICC edges presents a major task when applying static analysis[3].

3. RELATED WORK

This section introduces three of the notable previous work projects that Fordroid is based on.

3.1 Flowdroid

One of the inspirations for the Fordroid application was the static taint-analysis application **FlowDroid** [1]. Taint-analysis is an approach that follows the data flow (sensitive "tainted" information) within the application from some pre-defined source to its destination, the given "sink". This gives

exact information where the data goes and where it might be leaked. FlowDroid is used to detect privacy leaks, created by malicious intents or by carelessness, by analyzing the applications' bytecode and configuration files. FlowDroid is believed to be first of its type that correctly models Android lifecycle of applications, as well as handles callbacks and UI (input controls) widgets within the application. How it models these is by parsing the layout XML files, files containing executable codes, manifest file, etc. From the lifecycle and callback methods, it generates an Inter-procedural Control-Flow Graph (**ICFG**) through which it then traverses to track taints. Ending result are the all discovered flows in the shape of full path information, leading from sources to sinks which then help to minimize the overall number of missed leaks as well as false positives. The main con of FlowDroid is that it does not address ICC (Inter-Component Communication), it does not take into consideration that intents can be sent implicitly or explicitly between components.

3.2 Amandroid

Amandroid [5] is a general framework for conducting static analysis on Android, it introduces new approaches in solving the before explained challenges of conducting static analysis. Mainly, Amandroid successfully describes paths of data and control flow of all objects within an Android application across all application's components. The Amandroid Analysis Pipeline is given in figure 1 and is described in the following part of the section.

Section 2 states the different aspects of Android applications' architecture, one of which is that an application is not a closed system and depends largely on the core Android system. In order to conduct static analysis and to describe control flow in an application, analyzer needs an environment which models an application inside a system for that we then use Amandroid. Before modeling the Android environment, Amandroid first decompiles the APK¹ package which contains the source code, converts it with its own dex2IR translator to get the intermediate representation of application (**IR**), out of which it then builds an abstract syntax tree (**AST**). Amandroid follows the **Flowdroid**'s [1] steps in modeling the Android environment, but the key difference being that Amandroid models it on component-level contrary to Flowdroid's whole app-level.

Since Android is a component-based system and applications' components can activate system components, modeling the environment on component-level proves to be more accurate in control and data flow representation. After modeling the environment Amandroid manages to successfully handle Inter-Component communication, by finding the target- "sink" components. The destination of the ICC can be specified explicitly or implicitly in an outgoing intent. Destination of the explicit intent is often called by using Android system APIs which is easy to infer. The inferring of implicit intents Amandroid does by parsing the manifests of applications which specify intent filters for components. Amandroid then handles and builds Inter-procedural Control flow Graph (**ICFG**) and Inter-Component Data flow graph (**IDFG**). From **IDFG** Amandroid derives Data Dependence Graph

¹APK (Android package) - package file format used by the Android operating system for distribution and installation of mobile apps and middleware.

(DDG) which models instance and variable flow through the application. Building a precise Data Dependency Graph is of key importance in finding if an application leaks sensitive information, (i.e passwords, GPS coordinates). Given a source and a sink of information we can deduce if a given app leaks those kinds of information by analyzing the DDG.

Fordroid uses Amandroid graphs in order to analyze and find types of sensitive information written to local storage.

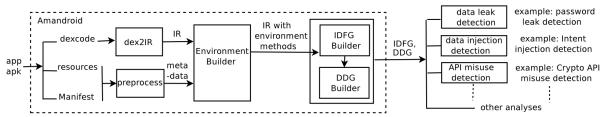


Figure 1: The Amandroid Analysis

3.3 Static Control Flow Analysis

Control-flow analysis [4] is a static code analysis technique used to determine the control flow of a program. This technique is essential for modeling the control flow of an application that is expressed as a control-flow graph. The way how Android platform interacts with the application code is through callbacks. So basically there are calls from the platform's event processing code to the relevant callback methods that are defined in the application code. Key aspect of the control flow analysis are the lifecycle and interaction of user-event-driven application components. Another part of this analysis is the representation of the all possible sequences of callbacks in a graph called Callback Control-Flow Graph (CCFG). The analysis of each callback method determines what other callbacks may be triggered next, and those are the information that provide the basis for the CCFG construction. The main goal of this technique is to determine all the valid paths.

4. FORDROID

4.1 Overview

Forensic analysis approaches are briefly described in section 1, in light of serious problems and shortcomings regarding scalability to a large number of applications, technical and equipment requirements of Dynamic analysis and Manual reverse engineering, static taint analysis has been proven to be more effective and globally more used. Application of the paper we are reviewing here, Fordroid, is an automated forensic tool that takes the static taint analysis approach in trying to give answers to What is the sensitive information stored (of which type), where is that information stored (i.e Paths to files or databases) and how the information is stored (i.e gives the structure of database tables) [6], because those are the most important questions asked when conducting a digital forensic analysis in mobile forensics.

4.2 Analysis approach

Fordroid is based on the Amandroid project which is described in section 3.2. Namely Fordroid uses Amandroid to decompile android application packages (APKs), AST, ICFG, IDFG, DDG and it also uses an API from Amandroid to find the types of information written in local storage [6]. In order

to find where the information is stored, Fordroid traverses the Data Dependency Graph from the taint sink until the path to a location where the information is written to local storage is found. Android provides many kinds of storage types for applications to store their data. Fordroid presents three different modules for finding where the information is stored by generalizing the storage types into the following three types:

- **Internal file storage** - Android system provides a private directory where applications store their private files in. Users and other applications cannot access these directories unless they have root access. When the application is uninstalled directory with its files is also removed from the system.
- **Shared Preferences** - Presents XML files with key-value pairs, which is used mainly when there is not a lot of data to be stored and is unstructured, each is managed by the framework and it can be shared or private.
- **Database storage** - Android stores structured data in the SQLite databases. Each database is isolated and accessible only to the applications it is created for.

4.2.1 Internal file storage

Writing to a file in Android can be done using three different native Java libraries, `FileWriter`, `BufferedWriter`, and `FileOutputStream`. Fordroid introduces their own algorithm for handling files, shown in algorithm 1 which handles all of the three different types of writing data to a file.

The algorithm first gets the method which writes sensitive information to a file as shown in line 3 of the algorithm 1. After getting the caller, "taint source", it then traverses the DDG and ICFG graphs in order to find caller's definition site as shown in line 4. After finding the definition site it simply parses first constructor of `FileWriter` (in this algorithm as a show-case `FileWriter` library is used) which is a file path. Handling the second constructor that takes two parameters, algorithm first gets the first parameter (the file object) and backtracks through ICFG and DDG graphs again to find its definition site and it then invokes function `getPathFromFile` shown in line 10 of algorithm 1.

As a result of invoking this routine Algorithm gets the number of parameters and and the value of the first parameter as shown in lines 13 and 14. If new `File()` accepts one parameter then the parameter is `pathToFile` as shown in line 15, if the number of parameters that `File()` is accepting is two then the second parameter is the latter part of the file path as shown in line 18. If the parameter returned from function `getPathFromFile` is actually a file object then the algorithm backtracks again through ICFG and DDG to find its definition and with function `getPathFromFile` that is shown on line 21 it gets the first part of the `FilePath` and simply concatenates it with `Para2` that the algorithm got from the first time invoked function `getPathFrom file`. As line 22 shows if the type of first parameter `Para1` is of type string then it represents the first part of file path which concatenated with `Para2` gives the full path of the file. Fordroid infers a path

to a file where "taint sink" writes information following the above-explained procedure.

Algorithm 1 Find_file_pat

```

1: Input: sk, ast, icfg, ddg;
2: Output: path;
3: fw = getCaller(sk, ast);
4: fw.def = defsite(fw, icfg, ddg);
5: para = getPara(fw.def, 1, ast);
6: if (type(para) == str) then
7:   path = para;
8: else if (type(para) == File) then
9:   para.def = defsite(para, ast);
10:  path = getPathFromFile(para.def, ast, icfg, ddg);
11: return path;
12: str getPathFromFile(f, ast, icfg, ddg){
13: num = numPara(f, ast);
14: para1 = getPara(f, 1, ast);
15: if (num == 1) then
16:   return para1;
17: else if (num == 2) then {
18:   para2 = getPara(f, 2, ast);
19:   if (type(para1) == File) then {
20:     para1.def = defsite(para1, icfg, ddg);
21:     return getPathFromFile(para1.def, ast, icfg, ddg) + para2;
22:   else if (type(para1) == str) then
23:     return para1 + para2; }}}

```

4.2.2 Shared preferences

For accessing existing SharedPrefrences files, there are three APIs that can be used: `getPreferences()`, `getDefaultSharedPreferences()` and `getSharedPreferences()`. The first two are used to find the default shared preference file that is being used in a specific context, while the last one is used to retrieve multiple shared preferences files defined strictly by the first parameter of the method, the file name, or path.

In Fordroid they have presented their own algorithm, algorithm 2, for finding paths that lead to SharedPrefrences file. Following the algorithm they have used, it can be seen it takes as an input a taint sink (location in the application where the sensitive information is stored), and the previously mentioned graphs, AST, ICFG and DDG, the output is the already said path to the SharedPrefrences file containing the looked for sensitive information. The algorithm's third line finds the editor of the taint sink by the help of the AST, and then backtracks through the ICFG and DDG graphs to find the definition location of the found editor. With this definition location, it again looks through the AST to find the wanted object of SharedPrefrences, the backtracking is repeated with this exact object through the ICFG and DDG graphs to find the exact location of the taint sink. Depending on the API that is used, the path is either gotten from the first parameter, or the default one is retrieved.

Algorithm 2 Shared_preferences

```

1: Input: sk, ast, icfg , ddg;
2: Output: path;
3: editor = getCaller(sk, ast);
4: ed_def = defsite(editor, icfg, ddg);
5: sp = getCaller(ed_def, ast);
6: sp_def = defsite(sp, icfg , ddg);
7: switch sp_def:
8:   case SharedPreferences:
9:     path = getPara(sp_def, 1, ast); break;
10:  default: path = defaultPath();
11: return path;

```

4.2.3 Database storage

The third module that Fordroid handles for storing data is database storage, as mentioned Android uses the SQLite databases for the storing. The algorithm that is used is the algorithm 3, and it again, as mentioned for algorithm 2, takes as an input a taint sink, and the AST, ICFG, and DDG. To know the exact location where the information is stored when databases are concerned, the exact database's name and the exact table name need to be found.

Table name is the first parameter of the taint sink, so that information is found immediately, as shown in the third line of algorithm 3. The second step is finding the caller (editor) of the taint sink, an SQLiteDatabase, through the AST, and afterwards, the ICFG and DDG are traversed backwards to find the definition location of this editor. The following step is looking again through the AST for the caller, named helper in the algorithm as seen in line 6, of this definition location, which is an extension of SQLiteOpenHelper (helper class for creation, opening and overall management of databases). To find the exact database name, the constructor of that helper is needed along with its invocation, both of which are found by the help of AST. The invocation of the constructor is the helper's superclass, and the second parameter of it contains the wanted database name.

Algorithm 3 DB_Storage

```

1: Input: sk, ast, icfg, ddg;
2: Output: <db_name, tb_name>;
3: tb.name = getPara(sk, 1, ast);
4: db = getCaller(sk, ast);
5: db.def = defsite(db, icfg, ddg);
6: helper = getCaller(db.def, ast);
7: cons = constructor(helper, ast);
8: db.name = getPara(supper, 2, ast);
9: return <db_name, tb.name>;

```

In order to find the structure of tables in databases Fordroid simply searches and monitors `execSQL()` API for executing SQL commands, Fordroid then searches for `CREATE TABLE` command and parses it. By parsing the `CREATE TABLE` command it gets table name and both types and names of each column in a table.

5. EVALUATION

For proving Fordroid's efficiency, the authors of the paper being reviewed, have tested it on randomly selected 100 applications from AppChina. Fordroid analysis took approximately 38 minutes for each application. The type categorization of the applications used in the analysis, as well as the number of components and APKS each category had, can be seen in table 1. The average number of components over all applications was 28.4, and the average number of paths per application was 23.1. These paths were determined by the taint analysis using all taint sinks found by Amandroid. The gotten results showed that Fordroid managed to find the precise locations of sensitive information for 98% of the paths. Out of the 100 applications they have tested it on, for 56% of them the conclusion was made they leak sensitive information, but only one-third of all write those to local storage. This sensitive information written to local storage are most often written in SharedPrefrences, out of 469 paths 416 were written in shared preferences, while sig-

category	APKs	comp.	time(min)	paths	paths to storage			where		APKs with paths	APKs write storage
					sp	db	file	suc.	fa.		
comm.	26	978	1827	310	27	0	7	33	1	11	6
entergame	26	278	207	422	196	18	8	221	1	11	8
newsinfo	24	715	902	360	30	4	10	38	6	16	10
tool	24	870	924	1221	163	0	6	166	3	18	12
total	100	2841	3860	2313	416	22	31	458	11	56	36
avg	/	28.4	38.6	23.1	41.6	2.2	3.1	45.8	1.1	/	/

Table 1: Results

nificantly less in files, 31, and then in the end are databases, with only 22 paths, as can be seen in table 1.

The exact structures of all of those databases were revealed with the help of Fordroid. The 2% of paths for which Fordroid was unsuccessful, was because of string operations and input dependencies. Some of those string operations was a combination of a hash operation and a substring operation over a specific resulting string, these kind of combinations are not supported in Fordroid, while the input dependencies were when one string depended on the input of a function being done in the background.

6. CONCLUSION

The proposed approach that is fully automated shows what, where and how sensitive information is stored in the local storage. It also overcomes the challenges related to the inter-component string propagation, string operations, and API invocations. While the manual digital forensic for Android applications is very time-consuming and hard to scale to a larger number of application, the proposed and implemented approach completed the analysis for 100 randomly selected applications in about 64h. The results obtained from the analyses had shown that with 98% rate *Fordroid* successfully located where the sensitive information is written to, its paths, and identifies the structure of all database tables which contain sensitive data.

7. REFERENCES

- [1] S. Arzt, S. Rasthofer, C. Fritz, E. Bodden, A. Bartel, J. Klein, Y. L. Traon, D. Octeau, and P. McDaniel. Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps. *ACM SIGPLAN Notices - PLDI '14*, 49(6):259–269, June 2014.
- [2] S. Dogan and E. Akbal. Analysis of mobile phones in digital forensics. *International Convention on Information and Communication Technology, Electronics and Microelectronics*, July 2017.
- [3] N. Scrivens and X. Lin. Android digital forensics: data, extraction and analysis. *ACM TUR-C '17*, May 2017.
- [4] H. W. Y. W. Shengqian Yang, Dacong Yan and A. Rountev. Static control-flow analysis of user-drivencallbacks in android applications. *IEEE*, August 2015.
- [5] O. X. Wei F., Roy S. Amandroid: a precise and general inter-component data flow analysis framework for security vetting of android apps. *Proc. CCS*, 2014.
- [6] T. Z. K. Y. F. W. Xiaodong Lin, Ting Chen. Automated forensic analysis of mobile applications on android devices. *Proc. DFRWS*, 2018.

Rekonstrukcija pretočene video vsebine iz predpomnilnika spletnega brskalnika Chrome

Martin Čebular

Fakulteta za računalništvo in
informatiko
Večna pot 113
1000, Ljubljana
mc0239@student.uni-lj.si

Blažka Blatnik

Fakulteta za računalništvo in
informatiko
Večna pot 113
1000, Ljubljana
bb3172@student.uni-lj.si

Jernej Černelč

Fakulteta za računalništvo in
informatiko
Večna pot 113
1000, Ljubljana
jc3731@student.uni-lj.si

Povzetek

Zmožnost ogleda video vsebin preko spletja je danes ena izmed najbolj zaželenih aktivnosti vsakodnevnega uporabnika svetovnega spletja. Današnji brskalniki omogočajo, da video vsebine za ogled ne potrebujemo v celoti prenašati na napravo, ampak so jo sposobni pretakati (angl. streaming), kar pomeni, da lahko s predvajanjem video vsebin pričnemo takoj, le-ta pa se v majhnih koščkih prenaša na napravo in se shranjuje v predpomnilnik (angl. cache) brskalnika. Z vidika forenzične analize je to zanimiv problem, ker predvajana in potencialno ogledana video vsebina ni bila nikoli neposredno prenesena na napravo. V datotečnem sistemu se datoteka z video vsebino ni nikoli nahajala v celoti, temveč je video vsebino potrebeno rekonstruirati iz večih datotek predpomnilnika brskalnika, v katerem je bila predvajana.

Ključne besede

predpomnenje, rekonstrukcija, digitalna forenzika, Google Chrome, Youtube, Facebook Live, Twitch

1. UVOD

V sklopu seminarske naloge smo analizirali članek z naslovom "Reconstructing Streamed Video Content: A Case Study on YouTube and Facebook Live Stream Content in the Chrome Web Browser Cache"[18].

Pretakanje (angl. streaming) video vsebine omogoča gledalcu, da lahko ob dostopu takoj začne s predvajanjem leta, brez predhodnega prenosa celotne datoteke z video vsebino. Namesto tega prenosa se med predvajanjem izvaja prednalaganje (angl. buffering), sprotno nalaganje manjših koščkov, ki predstavljajo manjše časovne segmente celotnega videoposnetka. Segmenti se nalagajo vnaprej in se na strani odjemalca (brskalnika) sestavljajo skupaj. Sestavljeni tako omogočajo tekoče predvajanje in ogled video vsebine. S forenzičnega vidika je pomembno dejstvo, da se za razliko od prenosa, ob pretakanju videa v datotečnem sistemu ne

ustvari enotna datoteka z video vsebino, temveč je pretočena video vsebina razdeljena na večje število manjših datotek, ki se nahajajo v predpomnilniku brskalnika.

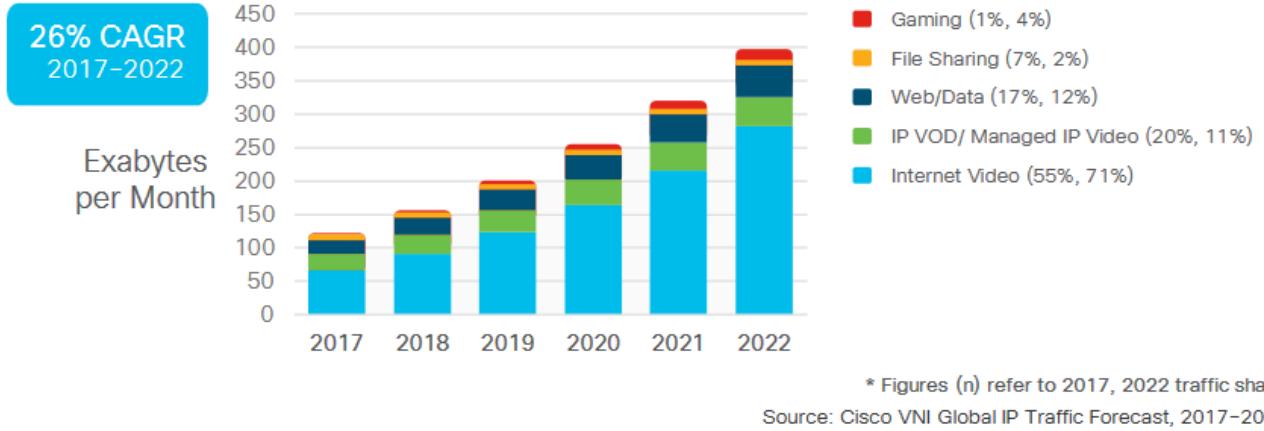
Poseben primer pretakanja video vsebine je pretakanje v živo (angl. live streaming). V tem primeru celotne video vsebine ni mogoče prenesti, ker se ta ustvarja sproti, v živo. Prav tako po končanem pretoku ponoven ogled vsebine ni več možen, razen v primeru, da uporabljeni platforma to omogoča. Kadar gre za primer ogleda takšne video vsebine, govorimo o klasičnem pretakanju in ne več o pretakanju v živo. Razlika pri pretakanju v živo je, da prednalaganje navadno ni prisotno, saj video vsebina še ni bila posnetna oziroma posredovana v platformo iz strani snemalca.

2. POMEN TEHNOLOGIJE PRETAKANJA VIDEA ZA DIGITALNO FORENZIKO

Tako pretočena kot prenesena video vsebina ima to lastnost, da je na voljo na določenem splettem naslovu. Če lahko iz podatkov o zgodovini brskanja ali iz koščkov video vsebine v predpomnilniku izluščimo naslov, od koder je bila video vsebina pretočena, lahko do spletnega naslova, in s tem video vsebine, dostopamo sami. To je koristno v primeru, ko nimamo vsebine videoposnetka. Vendar tu naletimo na dve težavi. Lahko se zgodi, da video vsebina na naslovu ni več dostopna (podobna težava se pojavi pri pretakanju v živo), poleg tega pa samo dostop do naslova z video vsebino ne dokazuje, da je bila video vsebina dejansko predvajana.

Nekoliko drugačna zgodba je pri pretakanju v živo. V tem primeru namreč dostopnost video vsebine ni zagotovljena in se lahko zgodi, da je od video vsebine ostalo samo to, kar se še nahaja v predpomnilniku brskalnika. Smo pa v primeru pretakanja v živo lahko dokaj prepričani, da je morebitna video vsebina iz predpomnilnika bila predvajana. Tu namreč lahko z veliko verjetnostjo izključimo možnost, da je bila vsebina samo naložena in ne tudi predvajana.

Pretakanje video vsebin postaja oziroma je že postal zelo pomemben del prometa v svetovnem spletu. Študija, ki jo je opravil Cisco [16], kaže da je v letu 2017 nekoliko več kot polovica (55%) prometa bila video vsebina, z napovedjo, da se bo ta odstotek povzpel na 71% do leta 2022 kot je prikazano na sliki 1. Od tega je v letu 2017 bilo 5% v živo pretočenih vsebin, z napovedjo povečanja na 17% v letu 2022.



Slika 1: Napovedi distribucije spletnega prometa med segmenti, kjer ima video promet več kot polovični delež in še raste [16].

Na žalost pa se tehnologija pretoka video vsebin izrablja za predvajanje in ogled nelegalnih vsebin. Od piratizacije do hujših kršitev, kot je distribucija otroške pornografije in video vsebin, ki nagovarjajo k nasilju ali terorizmu [8, 3, 6]. Dostopnost video vsebin je tako postala izjemno lahka, z vidika digitalne forenzične analize pa predstavlja izziv, kako odkriti ali je določena video vsebina bila pretočena in predvajana - kljub temu, da se nikoli ni oziroma se je za zelo kratek čas, nekoč nahajala v datotečnem sistemu odjemalca.

2.1 Pregled področja

Video posnetki in slike so pogosto uporabljeni kot dokazno gradivo na sodišču. Tako gradivo ima lahko različne izvore kot so posnetki nadzornih kamer, posnetki s pametnih telefonov, nizkokakovostni posnetki iz drugih naprav, in tako dalje. Ti posnetki lahko nosijo veliko bremenilnih ali oprostilnih dokazov. Za uporabo na sodišču je potrebno gradivu dokazati pristnost. Zanima nas, ali sta bila video oziroma slika naknadno spremenjeni, kar lahko zajema urejanje slik, spremicanje kompozicij, rezanje delov videa, s katero napravo sta bila slika oziroma video zajeta, kdo je avtor posnetka in kdo si je vsebino ogledal. Področje digitalne forenzične, ki se ukvarja s takšno analizo video posnetkov se imenuje video forenzika [20, 13].

3. ANALIZA PLATFORM ZA PRETAKANJE VIDEA

V tem poglavju bomo opisali nekaj nabolj uporabljenih platform za deljenje in pretakanje video vsebin.

3.1 Youtube

Youtube je platforma za deljenje in pretakanje video vsebin v lasti Google-a. Popularnost platforme potrjuje 184 milijonov uporabnikov samo iz Združenih držav [14], ki naj bi uporabljali njene storitve. Velika popularnost privablja tudi uporabnike, ki platformo zlorabljajo za širjenje sovraštva, pedofilije in terorizma [5, 7]. Za platformo na katero se vsako minuto prenese okoli 400 ur video vsebin [15], je



Slika 2: Rdeča puščica kaže na sivo območje, t.j. naložen in predpomnjen del videoposnetka.

omejevanje in nadzor teh vsebin velik zalogaj, zato ni presenečenje, da so tovrstne vsebine na platformi prisotne. Za dokazovanje krivde je potrebno dokazati, ne le da je osumljeni obiskal stran s sporno vsebino, pač pa, da si jo je tudi ogledal.

3.1.1 Povezava do videa in njegovo pretakanje

Povezava do vsakega videa, deljenega na platformi je podana s sledečo strukturo

<https://www.youtube.com/watch?v=<id>>

pri čemer <id> predstavlja enolični identifikator vsakega videa. Z analizo zgodovine brskalnika lahko določimo, če je uporabnik obiskal spletno stran z določeno video vsebino, ne moremo pa dokazati, da si je videoposnetek tudi ogledal. Prav tako je mogoče, da je bil video s podanim identifikatorjem že odstranjen. Če želimo dokazati kolikšen del videa si je uporabnik ogledal, tudi če je bil ta odstranjen, si lahko pomagamo z analizo predpomnilnika brskalnika.

DevTools - www.youtube.com/watch?v=ZUqzCsyE

Elements Console Memory Sources Network Performance Application Security Audits AdBlock

Filter Hide data URLs XHR JS CSS Img Media Font Doc WS Manifest Other

Name Headers Preview Response Timing

Request URL: http://r3---sn-uxahx5j1-aw01.googlevideo.com/videoplayback?<redacted>

Request Method: GET

Status Code: 200 OK

Remote Address: 185.72.60.78:443

Referrer Policy: origin-when-cross-origin

Response Headers

Accept-Ranges: bytes

Access-Control-Allow-Credentials: true

Access-Control-Allow-Origin: https://www.youtube.com

Access-Control-Expose-Headers: Client-Protocol, Content-Length, Content-Type, X-Bandwidth-Est, X-Bandwidth-Est2, X-Bandwidth-Est3, X-Bandwidth-App-Limited, X-Bandwidth-Est-App-Limited, X-Bandwidth-Est-Comp, X-Bandwidth-Avg, X-Head-Time-Millis, X-Head-Time-Sec, X-Head-Sequnum, X-Response-Itag, X-Restrict-Formats-Hint, X-Sequence-Num, X-Segment-Lmt, X-Walltime-Ms

Alt-Svc: quic=":443"; ma=2592000; v="46,44,43,39"

Cache-Control: private, max-age=20517

Connection: keep-alive

Content-Length: 644552

Content-Type: audio/webm

Date: Fri, 03 May 2019 15:38:12 GMT

Expires: Fri, 03 May 2019 15:38:12 GMT

Last-Modified: Fri, 07 Dec 2018 21:52:42 GMT

Server: gvs 1.0

Timing-Allow-Origin: https://www.youtube.com

X-Content-Type-Options: nosniff

Request Headers

Provisional headers are shown

Origin: https://www.youtube.com

Referer: https://www.youtube.com

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729.131 Safari/537.36

17 requests | 5.0 MB transferred | 5.0 MB resources

Console What's New

Slika 3: Na levi lahko opazimo klice za /videoplayback?, na desni pa vrednosti atributa Cache-Control.

General

Request URL: https://r3---sn-c0q1nly.googlevideo.com/videoplayback?<redacted>

Request Method: GET

Status Code: 200 OK

Remote Address: 74.125.154.8:443

Referrer Policy: origin-when-cross-origin

Response Headers

Accept-Ranges: bytes

Access-Control-Allow-Credentials: true

Access-Control-Allow-Origin: https://www.youtube.com

Access-Control-Expose-Headers: Client-Protocol, Content-Length, Content-Type, X-Bandwidth-Est, X-Bandwidth-Est2, X-Bandwidth-Est3, X-Bandwidth-App-Limited, X-Bandwidth-Est-App-Limited, X-Bandwidth-Est-Comp, X-Bandwidth-Avg, X-Head-Time-Millis, X-Head-Time-Sec, X-Head-Sequnum, X-Response-Itag, X-Restrict-Formats-Hint, X-Sequence-Num, X-Segment-Lmt, X-Walltime-Ms

Alt-Svc: quic=":443"; ma=2592000; v="46,44,43,39"

Cache-Control: private, max-age=20507

Connection: keep-alive

Content-Length: 1838201

Content-Type: video/webm

Date: Fri, 03 May 2019 15:38:22 GMT

Expires: Fri, 03 May 2019 15:38:22 GMT

Last-Modified: Fri, 07 Dec 2018 21:47:02 GMT

Server: gvs 1.0

Timing-Allow-Origin: https://www.youtube.com

X-Content-Type-Options: nosniff

General

Request URL: https://r3---sn-uxahx5j1-aw01.googlevideo.com/videoplayback?<redacted>

Request Method: GET

Status Code: 200 OK

Remote Address: 185.72.60.78:443

Referrer Policy: origin-when-cross-origin

Response Headers

Accept-Ranges: bytes

Access-Control-Allow-Credentials: true

Access-Control-Allow-Origin: https://www.youtube.com

Access-Control-Expose-Headers: Client-Protocol, Content-Length, Content-Type, X-Bandwidth-Est, X-Bandwidth-Est2, X-Bandwidth-Est3, X-Bandwidth-App-Limited, X-Bandwidth-Est-App-Limited, X-Bandwidth-Est-Comp, X-Bandwidth-Avg, X-Head-Time-Millis, X-Head-Time-Sec, X-Head-Sequnum, X-Response-Itag, X-Restrict-Formats-Hint, X-Sequence-Num, X-Segment-Lmt, X-Walltime-Ms

Alt-Svc: quic=":443"; ma=2592000; v="46,44,43,39"

Cache-Control: private, max-age=20517

Connection: keep-alive

Content-Length: 644552

Content-Type: audio/webm

Date: Fri, 03 May 2019 15:38:12 GMT

Expires: Fri, 03 May 2019 15:38:12 GMT

Last-Modified: Fri, 07 Dec 2018 21:52:42 GMT

Server: gvs 1.0

Timing-Allow-Origin: https://www.youtube.com

X-Content-Type-Options: nosniff

Slika 4: Primer dveh klicev na /videoplayback? iz katerih je razvidno, da se zvok in slika naložita ločeno.

```
https://r3--sn-uxax3vhna-aw0.e.googlevideo.com/videoplayback?id=o-AJS2zbFym-fYPh1Dz_qjkmYTTr4WOkMNcnGrCC3UC8&aitags=133%2C160%2C242%2C278&itag=242&source=youtube&requiressl=yes&mm=31%2C29&mn=sn-uxax3vhna-aw0e%2Csn-c0q7Inse&ms=au%2Crdlu&mv=m&pl=18&ei=JVTXLNLLlbj-gaOl6egCw&qcr=s&initcwndbps=531250&mime=video%2Fwebm&gir=yes&clen=2241230&dur=224.000&mt=1538082133035760&mt=1556829109&vjp=4&keepalive=yes&c=WEB&xcp=5532332&ip=89.143.105.251&ipbits=0&expire=1556850821&params=ip%2Cipbits%2Cexpire%2Cid%2Caitags%2Csource%2Crequiresl%2Cmm%2Cmn%2Cms%2Cmv%2Cpl%2CeI%2Cgcr%2Cinicwnbps%2Cmime%2Cgir%2Cclen%2Cdur%2Clmt&key=yt8&air=yes&signature=C220COA263009254B5F2B7A844AFB190DD1CA0EA.056B589F2FC4817B7BC61CB4E184A87F633C8FB0&cprn=GC-fOUQfQrGOKaP&cver=2.20190501&range=0-66472&altags=278&rn=796&rbuf=0
```

Slika 5: Primer URL-ja za prvi del videa.

3.1.2 Analiza predpomilnika

Da platforma Youtube uporablja predpomilnik lahko sklepamo iz sivega indikatorja na dnu videa. Slika 2 prikazuje video v prvih sekundah po začetku ogleda. Rdeč indikator kaže trenutno pozicijo v videu, medtem ko sivi indikator označuje prednaložen video, pripravljen za predvajanje.

Če prekinemo dostop do interneta, lahko opazimo, da so deli označeni s sivim indikatorjem še vedno na voljo za predvajanje. Iz tega lahko sklepamo, da se prednaloženi del shrani v brskalnikov predpomilnik.

Med predvajanjem videa lahko opazujemo promet na omrežju s pomočjo različnih orodij. Slika 3 prikazuje promet med predvajanjem video posnetka. Opazimo lahko, da se z vsakim uspešnim klicem na **/videoplayback?** sivi indikator premakne naprej, kar nakazuje, da je bil prednaložen dodaten del videa. Klici za **/videoplayback?** so prav tako edini, ki vsebino shranijo v predpomilnik, kar nakazuje pole "Cache-Control". Vrednost polja "Max-age" predstavlja koliko časa posamezni del ostane v predpomilniku v sekundah. V opazovanem primeru posamezni del videa v pomilniku ostane približno 5,7 ure.

Prav tako je zanimivo dejstvo, da se zvok in video prednaložita ločeno. To je razvidno iz slike 4. Pri obravnavi pomnilnika z uveljavljenimi forenzičnimi orodji, kot je na primer programska oprema "X-Ways forensics", so posamezni deli videa obravnavani kot samostojen video. Taka programska oprema zato ne rekonstruira videa kot celote. Rezultat je več manjših *.webm datotek, med katerimi je moč predvajati le eno [19].

3.1.3 Rekonstrukcija videa

Če želimo uspešno rekonstruirati celoten video je potrebno združiti vse *.webm datoteke, pridobljene iz predpomilnika. Pri tem je pomembno, da datoteke združujemo v pravilnem vrstnem redu. V nasprotnem primeru združenega videa ni mogoče predvajati. Vrstni red posameznih delov je mogoče določiti iz datuma prenosa oziroma iz pripadajočega URL-ja.

Iz URL-ja na sliki 5 je moč razbrati, da nam parameter **range** označuje pozicijo videa, kateri pripada preneseni del. V zgornjem primeru gre za prvi del videa, ki ga je mogoče tudi predvajati. Zanimivo je, da ob predvajjanju VLC predvajalnik kaže dolžino 22:07 čeprav se video preneha predvajati po približno 9 sekundah. Prikazani čas sovpada s parametrom **dur** iz zgornjega URL naslova. Končni video

```
https://video-amt2-1.xx.firebaseio.net/v/t42.1790-2/25083547_1981976162022480_6611541624600133632_n.mp4?_nc_cat=108&efg=eyJ2ZW5jb2RlX3RhZyl6ImRhc2hfjdNfNDI2X2NyZl8yM19tYWluXzMUf9mcnfNxzJYXVkaW8ifQ%3D%3D&_nc_ht=video-amt2-1.xx&o=h=3fd1a6e9fb93d9443a2fd3e8f0ac70eb&oe=5CCCCBC9&bytestart=1122&byteend=13538
```

Slika 6: Primer naslova pri prenosu video vsebine s Facebook-a.

je tako binarna konkatenacija posameznih delov v pravilnem vrstnem redu.

3.2 Facebook Live

Facebook Live je platforma za pretakanje video vsebin v živo in je ponujena kot del socialnega omrežja Facebook. Možnost snemanja in pretakanja video posnetka je na voljo vsem uporabnikom Facebook-a, medtem ko je dostop in predvajanje pretoka v živo odvisno od tega, ali je snemalec pretok nastavil kot javen ali zaseben. Pri tem so javno dostopni pretoki v živo dostopni tudi tistim, ki Facebook računa nimajo. Tako kot pri platformi YouTube, je tudi pri uporabi platforme Facebook Live prišlo do zlorab, med drugim pretkanje v živo video vsebin s spolnim nasiljem, izvajanjem splošnega nasilja in celo umorom [17, 1, 4].

Analiza predpomnjena pretoka v živo z uporabo platforme Facebook Live iz analiziranega članka je pokazala, da se pri predvajaju pretočenega video posnetka predpomnjene ne izvaja. Je pa po končanem pretakanju v živo video vsebina shranjena na strežniku in je do nje možno dostopati in jo ponovno predvajati - tu pa gre pravzaprav za navadno pretkanje video vsebine s Facebook-a in nič več pretkanje v živo.

3.2.1 Rekonstrukcija videa pretočenega s Facebooka
Rekonstrukcije se lotimo na podoben način kot pri rekonstrukciji videa, pretočenega s platforme YouTube (poglavlji 3.1.2 in 3.1.3). Manjše koščke videa povežemo z naslovi, s katerega so bili preneseni, te naslove analiziramo in koščke videa primerno uredimo po vrsti. Za razliko od pretoka videoposnetka iz YouTube-a, se v tem primeru slika in zvok preneseta skupaj, v eni datoteki.

Kot je že omenjeno v analiziranem članku, so tu ključni parametri **oe**, katerega vrednost je unikatni ID za videoposnetek, kjer vsi klici z isto vrednostjo **oe** pripadajo istemu pretoku videoposnetka, ter parametra **bytestart** in **byteend**, ki sta pomembna za pravilen vrstni red ob rekonstrukciji video posnetka iz manjših datotek.

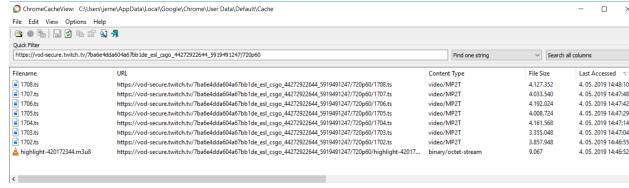
3.3 Twitch.tv

Twitch.tv je platforma za pretakanje video vsebin v živo ali za pregledovanje preteklih posnetkov. Večino njenega prometa predstavljajo vsebine splošnega igranja računalniških iger in prenosov raznih tekmovanj (eSports), vendar se trudi gostiti tudi dogodke iz vsakodnevnega življenja.

Ker je platforma odprta in dostopna vsakemu uporabniku z dostopom do spleta, so se v preteklosti in v sedanosti pojavile sporne vsebine, proti katerim se podjetje poskuša tem bolje boriti in jih v čim večji meri omejiti. Slednji primer na strani [21] pa pokaže, da je možno takšne vsebine

Name	Headers	Preview	Response	Timing
est_csgo.m3u8?allow_source=true&baking_bread=false...%22version%223A...				
CqxEej-FmejN2x0Wvhq0IB8oz6Loa69SEbCRG0wM1rCCEObYP..HEhDQu2B6u...	1 #EXTM3U			
CqxEej-FmejN2x0Wvhq0IB8oz6Loa69SEbCRG0wM1rCCEObYP..HEhDQu2B6u...	2 #EXT-X-TWITCH-INFO:NODE="video-edge-c67b94_1hr03",MANIFEST-NODE-TYPE="weaver_cluster",CLUSTER="1hr03",SUPPRESS="false",SERVER-TIME=...			
CqxEej-FmejN2x0Wvhq0IB8oz6Loa69SEbCRG0wM1rCCEObYP..HEhDQu2B6u...	3 #EXT-X-MEDIA:TYPE=VIDEO,GROUP-ID="chunked",NAME="1080p60_(source)",AUTOSELECT=YES,DEFAULT=YES			
CqxEej-FmejN2x0Wvhq0IB8oz6Loa69SEbCRG0wM1rCCEObYP..HEhDQu2B6u...	4 #EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=6956031,RESOLUTION=1920x1080,CODECS="avc1.64002A,mp4a.40.2",VIDEO="chunked",FRAME-RATE=60.0...			
CqxEej-FmejN2x0Wvhq0IB8oz6Loa69SEbCRG0wM1rCCEObYP..HEhDQu2B6u...	5 https://video-weaver.lhr03.hls.ttvnw.net/v1/playlist/CqxEej-FmejN2x0Wvhq0IB8oz6Loa69SEbCRG0wM1rCCEObYP..HEhDQu2B6u...			
CqxEej-FmejN2x0Wvhq0IB8oz6Loa69SEbCRG0wM1rCCEObYP..HEhDQu2B6u...	6 #EXT-X-MEDIA:TYPE=VIDEO,GROUP-ID="720p60",NAME="720p60",AUTOSELECT=YES,DEFAULT=YES			
CqxEej-FmejN2x0Wvhq0IB8oz6Loa69SEbCRG0wM1rCCEObYP..HEhDQu2B6u...	7 #EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=3462442,RESOLUTION=1280x720,CODECS="avc1.4D401F,mp4a.40.2",VIDEO="720p60",FRAME-RATE=60.0...			
CqxEej-FmejN2x0Wvhq0IB8oz6Loa69SEbCRG0wM1rCCEObYP..HEhDQu2B6u...	8 https://video-weaver.lhr03.hls.ttvnw.net/v1/playlist/CqxEej-FmejN2x0Wvhq0IB8oz6Loa69SEbCRG0wM1rCCEObYP..HEhDQu2B6u...			
CqxEej-FmejN2x0Wvhq0IB8oz6Loa69SEbCRG0wM1rCCEObYP..HEhDQu2B6u...	9 #EXT-X-MEDIA:TYPE=VIDEO,GROUP-ID="720p30",NAME="720p30",AUTOSELECT=YES,DEFAULT=YES			
CqxEej-FmejN2x0Wvhq0IB8oz6Loa69SEbCRG0wM1rCCEObYP..HEhDQu2B6u...	10 #EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=2412442,RESOLUTION=1280x720,CODECS="avc1.4D401F,mp4a.40.2",VIDEO="720p30",FRAME-RATE=30.0...			
CqxEej-FmejN2x0Wvhq0IB8oz6Loa69SEbCRG0wM1rCCEObYP..HEhDQu2B6u...	11 https://video-weaver.lhr03.hls.ttvnw.net/v1/playlist/CqxEej-FmejN2x0Wvhq0IB8oz6Loa69SEbCRG0wM1rCCEObYP..HEhDQu2B6u...			
CqxEej-FmejN2x0Wvhq0IB8oz6Loa69SEbCRG0wM1rCCEObYP..HEhDQu2B6u...	12 #EXT-X-MEDIA:TYPE=VIDEO,GROUP-ID="480p30",NAME="480p30",AUTOSELECT=YES,DEFAULT=YES			
CqxEej-FmejN2x0Wvhq0IB8oz6Loa69SEbCRG0wM1rCCEObYP..HEhDQu2B6u...	13 #EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=1467442,RESOLUTION=852x480,CODECS="avc1.4D401F,mp4a.40.2",VIDEO="480p30",FRAME-RATE=30.0...			
CqxEej-FmejN2x0Wvhq0IB8oz6Loa69SEbCRG0wM1rCCEObYP..HEhDQu2B6u...	14 https://video-weaver.lhr03.hls.ttvnw.net/v1/playlist/CqxEej-FmejN2x0Wvhq0IB8oz6Loa69SEbCRG0wM1rCCEObYP..HEhDQu2B6u...			
Name	Headers	Preview	Response	Timing
esi_csgo.m3u8?allow_source=true&baking_bread=false...%22version%223A...	1 #EXTM3U			
CqxEej-FmejN2x0Wvhq0IB8oz6Loa69SEbCRG0wM1rCCEObYP..HEhDQu2B6u...	2 #EXT-X-MEDIA:TYPE=VIDEO,GROUP-ID="chunked",NAME="1080p60_(source)",AUTOSELECT=YES,DEFAULT=YES			
CqxEej-FmejN2x0Wvhq0IB8oz6Loa69SEbCRG0wM1rCCEObYP..HEhDQu2B6u...	3 #EXT-X-TARGETDURATION:6			
CqxEej-FmejN2x0Wvhq0IB8oz6Loa69SEbCRG0wM1rCCEObYP..HEhDQu2B6u...	4 #EXT-X-MEDIA:SEQUENCE:12639			
CqxEej-FmejN2x0Wvhq0IB8oz6Loa69SEbCRG0wM1rCCEObYP..HEhDQu2B6u...	5 #EXT-X-TWITCH-ELAPSED-SECS:25303.379			
CqxEej-FmejN2x0Wvhq0IB8oz6Loa69SEbCRG0wM1rCCEObYP..HEhDQu2B6u...	6 #EXT-X-TWITCH-TOTAL-SECS:25333.409			
CqxEej-FmejN2x0Wvhq0IB8oz6Loa69SEbCRG0wM1rCCEObYP..HEhDQu2B6u...	7 #EXT-X-PROGRAM-DATE-TIME:2019-05-04T11:00:35.525Z			
CqxEej-FmejN2x0Wvhq0IB8oz6Loa69SEbCRG0wM1rCCEObYP..HEhDQu2B6u...	8 #EXTINF-2,092, live			
CqxEej-FmejN2x0Wvhq0IB8oz6Loa69SEbCRG0wM1rCCEObYP..HEhDQu2B6u...	9 https://video-edge-c67b94_1hr03.abs.hls.ttvnw.net/v1/segment/CvED0znC0iWw0c_9PliajIYeUNGCuwnPj_34gwHTaVImYpsfk13N0iVaDh0iKQwmwHpw5fr...			
CqxEej-FmejN2x0Wvhq0IB8oz6Loa69SEbCRG0wM1rCCEObYP..HEhDQu2B6u...	10 #EXT-X-PROGRAM-DATE-TIME:2019-05-04T11:00:37.527Z			
CqxEej-FmejN2x0Wvhq0IB8oz6Loa69SEbCRG0wM1rCCEObYP..HEhDQu2B6u...	11 #EXTINF-2,092, live			
CqxEej-FmejN2x0Wvhq0IB8oz6Loa69SEbCRG0wM1rCCEObYP..HEhDQu2B6u...	12 https://video-edge-c67b94_1hr03.abs.hls.ttvnw.net/v1/segment/CvEdh8KIna-97fn95XpxC_miW9m8011Qm4V0OLUcggV7xs:06e0j7nu0cYcQQFLXRbh0tE8...			
CqxEej-FmejN2x0Wvhq0IB8oz6Loa69SEbCRG0wM1rCCEObYP..HEhDQu2B6u...	13 #EXT-X-PROGRAM-DATE-TIME:2019-05-04T11:00:39.529Z			
CqxEej-FmejN2x0Wvhq0IB8oz6Loa69SEbCRG0wM1rCCEObYP..HEhDQu2B6u...	14 #EXTINF-2,092, live			

Slika 7: Primer vsebine datoteke *.m3u8 z URL-ji do želene vsebine, zgoraj. Primer vsebine datoteke *.m3u8 z URL-ji do segmentov posnetka, spodaj.



Slika 8: Prikaz brskalniškega predpomnilnika.

uporabiti kot dokazno gradivo na sodišču. Naša naloga je bilo analizirati, ali je možno takšno vsebino rekonstruirati iz predpomnilnika brskalnika nekoga, ki si je posnetek ogledoval v živo.

3.3.1 Pretakanje vsebin na Twitch.tv

Twitch.tv za pretakanje video vsebin uporablja protokol HLS s podporo nekaj ostalih protokolov [9]. Najprej pridobimo datoteko *.m3u8 iz katere izberemo želeni URL za vsebino katero si želimo ogledati v živo. Iz slike 7 zgoraj je razvidno, da imamo različne povezave za različne resolucije. Iz tega URL-ja nato pridobimo novo datoteko *.m3u8, ki vsebuje URL-je do segmentov (datoteke s končnico .ts), ki predstavljajo dele posnetka kot je prikazano na sliki 7 spodaj. Kličemo jih v pravilnem vrstnem redu in pridobimo trenutni posnetek v živo. Ko pokličemo vse URL-je, ponovno kličemo URL izbrane vsebine, ki vrne posodobljene URL-je segmentov.

3.3.2 Rekonstrukcija videa pretočenega s Twitch.tv-ja

Izkazalo se je da se video vsebina kadar si uporabnik ogleduje posnetek v živo v predpomnilnik ne shranjuje saj je polje, ki označuje ali se podatki shranjujejo v predpomnilnik nastavljen na negativno.

Pri predvajanju prednaloženega posnetka pa se z razliko od prenosa v živo, posnetki shranjujejo v brskalniški predpomnilnik. To smo razbrali tako s pomočjo programa Chrome-CacheView kot je razvidno iz slike 8 in nato še z analizo protoma, kjer smo opazili da je polje Cache-control v odgovoru

prenosnih paketov nastavljeneno na resnično. Tako imamo na voljo * ts datoteke, ki jih lahko po številčnem redu sestavimo skupaj in pridobimo celoten posnetek skupaj z zvokom.

4. DELOVANJE PREDPOMNILNIKA V BRSKALNIKU GOOGLE CHROME

Opis delovanja predpomnilnika v nadaljevanju temelji na brskalniku Google Chrome, verziji 74.0.3729.131, ki je bila najnovejša v času pisanja članka. V analiziranem članku je bila uporabljena verzija 63.0.3239.132. Ugotovili smo, da se delovanje predpomnenja med verzijami ni spremenilo. Implementacija predpomnenja pri brskalniku Google Chrome je odvisna od operacijskega sistema. Predpomnenje na sistemih Windows in Mac OS se izvaja drugače kot na sistemih Linux, vključno z Android [12].

V analiziranem članku je bila raziskava izvedena na operacijskem sistemu Windows 10. Predpomnilnik brskalnika Chrome se nahaja v direktoriju:

C:\Users\<user>\AppData\Local\Google\Chrome\UserData\<chrome-profile>\Cache,

kjer je <user> uporabniško ime v operacijskem sistemu Windows, <chrome-profile> pa direktorij profila v brskalniku Google Chrome. Brskalnik omogoča nastavitev več profilov oziroma uporabnikov, pri čemer ima vsak ločene podatke o brskanju. Privzeta vrednost <chrome-profile> je "Default", za vse nadaljnje uporabnike pa "Profile 1", "Profile 2", itd...

Predpomnilnik na spletnem brskalniku Google Chrome je implementiran tako, da se ne veča neomejeno, ampak je omejen z določeno velikostjo in ima algoritmom, ki skrbi za brisanje starejših vnosov [11]. Iz forenzičnega vidika je to slabo lastnost, saj pomeni brisanje starejših dokazov oziroma mora biti preiskava narejena preden je uporabnik dlje časa po tarčni vsebini uporabljal brskalnik. Predpomnilnik je prav tako odporen na izpade brskalnika, vendar izpad gostiteljskega računalnika povzroči okvaro celotnega predpomnilnika.

ChromeCacheView C:\Users\Blazka\AppData\Local\Google\Chrome\User Data\Profile 1\Cache

File Edit View Options Help

Quick Filter: id=o-AGYogDjeQVAzW798pjsdMswUKH5UchX3NSTo6YdvL2&aitags=133%2.webm

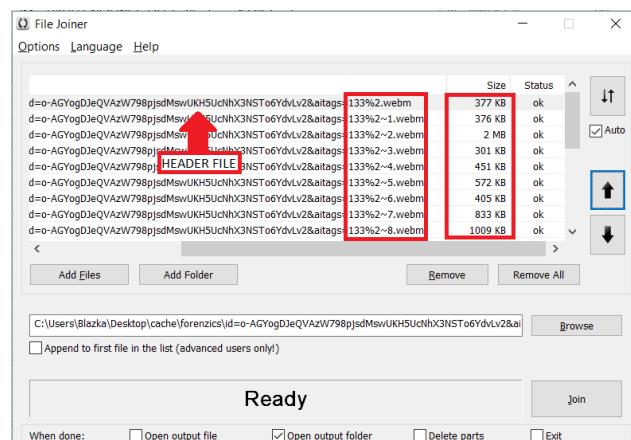
Find one string Search all columns

Filename	URL	Content ...	File Size	Last Accessed	Server Time	Server Last Mo...	Expire Time	Server Name ^
id=o-AGYogDjeQVAzW798pjsdMswUKH5UchX3NSTo6YdvL2&aitags=133%2.webm	https://r3---sn-uxaxh5j-aw0.googlevideo.com/videoplayback?id=...	audio/webm	644,552	3.05.2019 17:...	3.05.2019 17:38:12	7.12.2018 22:...	3.05.2019 17:...	gvs 1.0
id=o-AGYogDjeQVAzW798pjsdMswUKH5UchX3NSTo6YdvL2&aitags=133%2.webm	https://r3---sn-uxaxh5j-aw0.googlevideo.com/videoplayback?id=...	audio/webm	615,360	3.05.2019 17:...	3.05.2019 17:26:36	7.12.2018 22:...	3.05.2019 17:...	gvs 1.0
id=o-AGYogDjeQVAzW798pjsdMswUKH5UchX3NSTo6YdvL2&aitags=133%2.webm	https://r3---sn-uxaxh5j-aw0.googlevideo.com/videoplayback?id=...	audio/webm	412,078	3.05.2019 17:...	3.05.2019 17:25:16	7.12.2018 22:...	3.05.2019 17:...	gvs 1.0
id=o-AGYogDjeQVAzW798pjsdMswUKH5UchX3NSTo6YdvL2&aitags=133%2.webm	https://r3---sn-uxaxh5j-aw0.googlevideo.com/videoplayback?id=...	audio/webm	65,536	3.05.2019 17:...	3.05.2019 17:25:09	7.12.2018 22:...	3.05.2019 17:...	gvs 1.0
id=o-AGYogDjeQVAzW798pjsdMswUKH5UchX3NSTo6YdvL2&aitags=133%2.webm	https://r3---sn-uxaxh5j-aw0.googlevideo.com/videoplayback?id=...	audio/webm	213,951	3.05.2019 17:...	3.05.2019 17:25:10	7.12.2018 22:...	3.05.2019 17:...	gvs 1.0
id=o-AGYogDjeQVAzW798pjsdMswUKH5UchX3NSTo6YdvL2&aitags=133%2.webm	https://r3---sn-uxaxh5j-aw0.googlevideo.com/videoplayback?id=...	audio/webm	100,315	3.05.2019 17:...	3.05.2019 17:25:09	7.12.2018 22:...	3.05.2019 17:...	gvs 1.0
id=o-AGYogDjeQVAzW798pjsdMswUKH5UchX3NSTo6YdvL2&aitags=133%2.webm	https://r3---sn-uxaxh5j-aw0.googlevideo.com/videoplayback?id=...	audio/webm	620,689	3.05.2019 17:...	3.05.2019 17:25:39	7.12.2018 22:...	3.05.2019 17:...	gvs 1.0
id=o-AGYogDjeQVAzW798pjsdMswUKH5UchX3NSTo6YdvL2&aitags=133%2.webm	https://r3---sn-uxaxh5j-aw0.googlevideo.com/videoplayback?id=...	audio/webm	603,327	3.05.2019 17:...	3.05.2019 17:26:09	7.12.2018 22:...	3.05.2019 17:...	gvs 1.0
id=o-AGYogDjeQVAzW798pjsdMswUKH5UchX3NSTo6YdvL2&aitags=133%2.webm	https://r3---sn-uxaxh5j-aw0.googlevideo.com/videoplayback?id=...	audio/webm	612,721	3.05.2019 17:...	3.05.2019 17:38:42	7.12.2018 22:...	3.05.2019 17:...	gvs 1.0
id=o-AGYogDjeQVAzW798pjsdMswUKH5UchX3NSTo6YdvL2&aitags=133%2.webm	https://r3---sn-uxaxh5j-aw0.googlevideo.com/videoplayback?id=...	audio/webm	66,353	3.05.2019 17:...	3.05.2019 17:25:09	7.12.2018 22:...	3.05.2019 17:...	gvs 1.0
id=o-AGYogDjeQVAzW798pjsdMswUKH5UchX3NSTo6YdvL2&aitags=133%2.txt	https://r3---sn-uxaxh5j-aw0.googlevideo.com/videoplayback?id=...	text/plain	804	3.05.2019 17:...	3.05.2019 17:25:21	2.05.2007 12:...	3.05.2019 17:...	gvs 1.0
id=o-AGYogDjeQVAzW798pjsdMswUKH5UchX3NSTo6YdvL2&aitags=133%2.txt	https://r3---sn-uxaxh5j-aw0.googlevideo.com/videoplayback?id=...	text/plain	826	3.05.2019 17:...	3.05.2019 17:25:09	2.05.2007 12:...	3.05.2019 17:...	gvs 1.0
id=o-AGYogDjeQVAzW798pjsdMswUKH5UchX3NSTo6YdvL2&aitags=133%2.webm	https://r3---sn-cq7iInly.googlevideo.com/videoplayback?id=o-AGY...	video/webm	585,205	3.05.2019 17:...	3.05.2019 17:25:30	7.12.2018 22:...	3.05.2019 17:...	gvs 1.0
id=o-AGYogDjeQVAzW798pjsdMswUKH5UchX3NSTo6YdvL2&aitags=133%2.webm	https://r3---sn-cq7iInly.googlevideo.com/videoplayback?id=o-AGY...	video/webm	308,532	3.05.2019 17:...	3.05.2019 17:25:19	7.12.2018 22:...	3.05.2019 17:...	gvs 1.0
id=o-AGYogDjeQVAzW798pjsdMswUKH5UchX3NSTo6YdvL2&aitags=133%2.webm	https://r3---sn-cq7iInly.googlevideo.com/videoplayback?id=o-AGY...	video/webm	1,838,201	3.05.2019 17:...	3.05.2019 17:38:22	7.12.2018 22:...	3.05.2019 17:...	gvs 1.0
id=o-AGYogDjeQVAzW798pjsdMswUKH5UchX3NSTo6YdvL2&aitags=133%2.webm	https://r3---sn-cq7iInly.googlevideo.com/videoplayback?id=o-AGY...	video/webm	293,719	3.05.2019 17:...	3.05.2019 17:26:19	7.12.2018 22:...	3.05.2019 17:...	gvs 1.0
id=o-AGYogDjeQVAzW798pjsdMswUKH5UchX3NSTo6YdvL2&aitags=133%2.webm	https://r3---sn-cq7iInly.googlevideo.com/videoplayback?id=o-AGY...	video/webm	995,706	3.05.2019 17:...	3.05.2019 17:26:09	7.12.2018 22:...	3.05.2019 17:...	gvs 1.0
id=o-AGYogDjeQVAzW798pjsdMswUKH5UchX3NSTo6YdvL2&aitags=133%2.webm	https://r3---sn-cq7iInly.googlevideo.com/videoplayback?id=o-AGY...	video/webm	580,561	3.05.2019 17:...	3.05.2019 17:26:29	7.12.2018 22:...	3.05.2019 17:...	gvs 1.0
id=o-AGYogDjeQVAzW798pjsdMswUKH5UchX3NSTo6YdvL2&aitags=133%2.webm	https://r3---sn-cq7iInly.googlevideo.com/videoplayback?id=o-AGY...	video/webm	2,040,518	3.05.2019 17:...	3.05.2019 17:25:10	7.12.2018 22:...	3.05.2019 17:...	gvs 1.0
id=o-AGYogDjeQVAzW798pjsdMswUKH5UchX3NSTo6YdvL2&aitags=133%2.webm	https://r3---sn-cq7iInly.googlevideo.com/videoplayback?id=o-AGY...	video/webm	2,021,310	3.05.2019 17:...	3.05.2019 17:27:00	7.12.2018 22:...	3.05.2019 17:...	gvs 1.0
id=o-AGYogDjeQVAzW798pjsdMswUKH5UchX3NSTo6YdvL2&aitags=133%2.webm	https://r3---sn-cq7iInly.googlevideo.com/videoplayback?id=o-AGY...	video/webm	668,130	3.05.2019 17:...	3.05.2019 17:25:59	7.12.2018 22:...	3.05.2019 17:...	gvs 1.0
id=o-AGYogDjeQVAzW798pjsdMswUKH5UchX3NSTo6YdvL2&aitags=133%2.webm	https://r3---sn-cq7iInly.googlevideo.com/videoplayback?id=o-AGY...	video/webm	1,033,709	3.05.2019 17:...	3.05.2019 17:25:49	7.12.2018 22:...	3.05.2019 17:...	gvs 1.0
id=o-AGYogDjeQVAzW798pjsdMswUKH5UchX3NSTo6YdvL2&aitags=133%2.webm	https://r3---sn-cq7iInly.googlevideo.com/videoplayback?id=o-AGY...	video/webm	1,265	3.05.2019 17:...	3.05.2019 17:25:16	7.12.2018 22:...	3.05.2019 17:...	gvs 1.0
id=o-AGYogDjeQVAzW798pjsdMswUKH5UchX3NSTo6YdvL2&aitags=133%2.webm	https://r3---sn-cq7iInly.googlevideo.com/videoplayback?id=o-AGY...	video/webm	1,561,905	3.05.2019 17:...	3.05.2019 17:26:36	7.12.2018 22:...	3.05.2019 17:...	gvs 1.0
id=o-AGYogDjeQVAzW798pjsdMswUKH5UchX3NSTo6YdvL2&aitags=133%2.webm	https://r3---sn-cq7iInly.googlevideo.com/videoplayback?id=o-AGY...	video/webm	1,014,584	3.05.2019 17:...	3.05.2019 17:25:50	7.12.2018 22:...	3.05.2019 17:...	gvs 1.0
id=o-AGYogDjeQVAzW798pjsdMswUKH5UchX3NSTo6YdvL2&aitags=133%2.webm	https://r3---sn-cq7iInly.googlevideo.com/videoplayback?id=o-AGY...	video/webm	2,052,511	3.05.2019 17:...	3.05.2019 17:38:49	7.12.2018 22:...	3.05.2019 17:...	gvs 1.0
id=o-AGYogDjeQVAzW798pjsdMswUKH5UchX3NSTo6YdvL2&aitags=133%2.webm	https://r3---sn-cq7iInly.googlevideo.com/videoplayback?id=o-AGY...	video/webm	1,265	3.05.2019 17:...	3.05.2019 17:25:21	7.12.2018 22:...	3.05.2019 17:...	gvs 1.0
id=o-AGYogDjeQVAzW798pjsdMswUKH5UchX3NSTo6YdvL2&aitags=133%2.webm	https://r3---sn-cq7iInly.googlevideo.com/videoplayback?id=o-AGY...	video/webm	385,112	3.05.2019 17:...	3.05.2019 17:25:09	7.12.2018 22:...	3.05.2019 17:...	gvs 1.0
id=o-AGYogDjeQVAzW798pjsdMswUKH5UchX3NSTo6YdvL2&aitags=133%2.webm	https://r3---sn-cq7iInly.googlevideo.com/videoplayback?id=o-AGY...	video/webm	386,408	3.05.2019 17:...	3.05.2019 17:25:09	7.12.2018 22:...	3.05.2019 17:...	gvs 1.0
id=o-AGYogDjeQVAzW798pjsdMswUKH5UchX3NSTo6YdvL2&aitags=133%2.webm	https://r3---sn-cq7iInly.googlevideo.com/videoplayback?id=o-AGY...	video/webm	461,482	3.05.2019 17:...	3.05.2019 17:25:29	7.12.2018 22:...	3.05.2019 17:...	gvs 1.0
id=o-AGYogDjeQVAzW798pjsdMswUKH5UchX3NSTo6YdvL2&aitags=133%2.webm	https://r3---sn-cq7iInly.googlevideo.com/videoplayback?id=o-AGY...	video/webm	421,301	3.05.2019 17:...	3.05.2019 17:26:09	7.12.2018 22:...	3.05.2019 17:...	gvs 1.0
id=o-AGYogDjeQVAzW798pjsdMswUKH5UchX3NSTo6YdvL2&aitags=133%2.webm	https://r3---sn-cq7iInly.googlevideo.com/videoplayback?id=o-AGY...	video/webm	527,014	3.05.2019 17:...	3.05.2019 17:26:19	7.12.2018 22:...	3.05.2019 17:...	gvs 1.0
id=o-AGYogDjeQVAzW798pjsdMswUKH5UchX3NSTo6YdvL2&aitags=133%2.webm	https://r3---sn-cq7iInly.googlevideo.com/videoplayback?id=o-AGY...	video/webm	414,996	3.05.2019 17:...	3.05.2019 17:25:39	7.12.2018 22:...	3.05.2019 17:...	gvs 1.0

36 item(s), 1 Selected (0.81 KB)

NuSoft Freeware. <http://www.nusoft.net>

Slika 9: Pridobljene datoteke za podani identifikator posnetka. Zeleno polje označuje zvočne datoteke (audio/webm), rumeno polje označuje tekstovne datoteke (text/plain) in vijolično polje označuje video datoteke (video/webm).

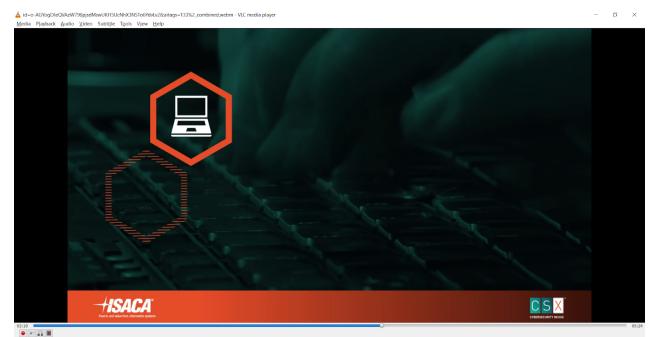


Slika 10: Urejene .webm datoteke pridobljene iz Google Chrome predpomnilnika, naložene v program File Joiner pripravljene za združitev.

Za rekonstrukcijo vsebine (tako videa kot tudi druge, npr. zvok, slike) v predpomnilniku je bil v analiziranem članku uporabljen program ChromeCacheView [10].

5. REKONSTRUKCIJA VIDEA

Za rekonstrukcijo videa smo izbrali platformo Youtube, brskalnik Google Chrome verzije 74.0.3729.131 in operacijski sistem Windows 10. Rekonstruirati smo želeli video, ki smo ga delno že pogledali [2]. Pred ogledom videa predpomnil-



Slika 11: Prikaz predvajanja rekonstruiranega videa.

nika nismo brisali, prav tako po ogledu videa nismo odklopili računalnika od omrežja, saj smo želeli video rekonstruirati v čim bolj realnih okoliščinah.

V prvem koraku smo pridobili vsebino Google Chrome predpomnilnika s programom ChromeCacheViewer. Ta nam vrne 6201 elementov v skupni velikosti 182,37 MB. Z uporabo hitrega filtra (angl. Quick Filter) poiščemo samo elemente, ki vsebujejo identifikator našega posnetka, pridobljen iz analize omrežja. Podan identifikator mora biti prisoten v vseh URL-jih, ki se navezujejo na isti video. Slika 9 prikazuje pridobljene datoteke za podani identifikator. Opazimo lahko, da posamezen video sestavlja deli zvoka, deli videa in tekstovne datoteke, v skupni velikosti 22,21 MB.

Posamezne datoteke formata “video/webm” uredimo glede na parameter “range”. Poiščemo datoteko, ki ima začetek “range” parametra enako 0. To je naša začetna datoteka in hkrati edina video datoteka iz predpomnilnika, ki jo je mogoče predvajati brez posebne obdelave. Nadaljujemo z iskanjem datoteke glede na naraščanje “video/webm” parametra. Z uporabo programa za konkatenacijo binarnih datotek združimo datoteke v urejenem vrstnem redu, kar je prikazano na sliki 10.

Rezultat je prvoten video, ki se brez težav zažene v media predvajalniku VLC in kaže dolžino 05:24, kar je dejanska dolžina izvormega video posnetka. Video je velik 18,879 KB, predvajanje pa se konča pri času 3:14 kar pomeni, da se video ni presenet v celoti. Iz tega lahko sklepamo (če predpostavimo, da uporabnik vsebine pomnilnika ni spremjal), da uporabnik videa ni nikoli pogledal do konca.

Kot že omenjeno se slika in zvok videa prenašata ločeno. Posledično rekonstruiran video nima zvoka. Predvajanje rekonstruiranega videa z predvajalnikom VLC prikazuje slika 11.

6. ZAKLJUČEK

V raziskavi članka smo razširili nekatere teme in tudi sami izvedli rekonstrukcijo video posnetka iz predpomnilnika brskalnika Google Chrome. Ugotovili smo, da pridobitev posnetkov, ki so bili ogledani na različnih platformah ne zahteva veliko korakov. Rekonstrukcija pa je možna le ob pravilni nastavitevi paketov, da se le ti predpomnijo na odjemalcu. Raziskavo bi lahko v prihodnjem delu uporabili kot osnovo za realno forenzično raziskavo in ocenili uporabnost različnih pristopov.

7. LITERATURA

- [1] Facebook live killer says he is looking for victims. Dosegljivo: <https://www.bbc.com/news/av/world-us-canada-39617888/facebook-live-killer-says-he-is-looking-for-victims>, 2017. [Dostopano: 30. 4. 2019].
- [2] Overview of digital forensics. Dosegljivo: https://www.youtube.com/watch?v=ZUqzcQc_syE, 2017. [Dostopano: 30. 4. 2019].
- [3] Sadiq khan urges youtube to remove ‘violent gang culture videos’. Dosegljivo: <https://www.bbc.com/news/uk-england-london-40849611>, 2017. [Dostopano: 1. 5. 2019].
- [4] Thai man kills baby on facebook live then takes own life. Dosegljivo: <https://www.bbc.com/news/world-asia-39706205>, 2017. [Dostopano: 30. 4. 2019].
- [5] Youtube child abuse reporting system ‘flawed’. Dosegljivo: <https://www.bbc.com/news/av/stories-42105526/youtube-child-abuse-reporting-system-flawed>, 2017. [Dostopano: 30. 4. 2019].
- [6] Youtube removes dead extremist’s videos. Dosegljivo: <https://www.bbc.com/news/technology-41969461>, 2017. [Dostopano: 1. 5. 2019].
- [7] Youtube to restrict ‘disturbing’ children’s videos, if flagged. Dosegljivo: <https://www.bbc.com/news/technology-41942306>, 2017. [Dostopano: 30. 4. 2019].
- [8] Man pleads guilty over ‘widely shared’ abuse video. Dosegljivo: <https://www.bbc.com/news/uk-england-42639072>, 2018. [Dostopano: 1. 5. 2019].
- [9] Twitch. Dosegljivo: <https://inspect.cool/2018/08/31/twitch/#video-streaming>, 2018. [Dostopano: 2. 5. 2019].
- [10] Chromecacheview v1.86. Dosegljivo: http://www.nirsoft.net/utils/chrome_cache_view.html, 2019. [Dostopano: 28. 4. 2019].
- [11] Disk cache. Dosegljivo: <https://www.chromium.org/developers/design-documents/network-stack/disk-cache>, 2019. [Dostopano: 3. 5. 2019].
- [12] disk cache api. Dosegljivo: https://github.com/chromium/chromium/tree/master/net/disk_cache#the-implementations, 2019. [Dostopano: 30. 4. 2019].
- [13] Forensic video analysis. Dosegljivo: https://en.wikipedia.org/wiki/Forensic_video_analysis, 2019. [Dostopano: 2. 5. 2019].
- [14] Which countries watch the most youtube? Dosegljivo: <https://www.worldatlas.com/articles/which-countries-watch-the-most-youtube.html>, 2019. [Dostopano: 1. 5. 2019].
- [15] Youtube statistics – 2019. Dosegljivo: <https://merchdope.com/youtube-stats/>, 2019. [Dostopano: 2. 5. 2019].
- [16] V. Cisco. Cisco visual networking index: Forecast and trends, 2017–2022. *White Paper*, 2018.
- [17] G. Dunlop. The bruising clash over a facebook live stream. Dosegljivo: <https://www.bbc.com/news/world-australia-38876428>, 2017. [Dostopano: 30. 4. 2019].
- [18] G. Horsman. Reconstructing streamed video content: A case study on youtube and facebook live stream content in the chrome web browser cache. *Digital Investigation*, 26:S30–S37, 2018.
- [19] G. Horsman. Reconstructing streamed video content: A case study on youtube and facebook live stream content in the chrome web browser cache. *Digital Investigation*, 26:S30 – S37, 2018.
- [20] S. Milani, M. Fontani, P. Bestagini, M. Barni, A. Piva, M. Tagliasacchi, and S. Tubaro. An overview on video forensics. *APSIPA Transactions on Signal and Information Processing*, 1:e2, 2012.
- [21] V. Molyneux and C. Whitten. Thai man kills baby on facebook live then takes own life. Dosegljivo: <https://www.newshub.co.nz/home/world/2018/12/twitch-streamer-arrested-after-livestreaming-himself-abusing-wife.html>, 2018. [Dostopano: 2. 5. 2019].

The Growing Impact of IoT on Digital Forensics*

Nejc Nadizar
Faculty of Mathematics and Physics
Faculty of Computer and Information Science
nn4669@student.uni-lj.si

ABSTRACT

Internet of Things (*IoT*) strives to create a highly heterogeneous network of interconnected nodes that autonomously communicate with each other. Although the nature of devices can be directly related with conventional digital forensics, *IoT* brings a number of challenges that add additional complexity to the existing digital forensic investigations. In this paper we overview the field of *IoT* and its difference from traditional (conventional) digital forensics. We also break down different types of devices, define which devices fall into each category, their hardware composition and what kind of data we can gather from them. Finally we look at the number of challenges that field brings or change and provide brief overview for each one.

Keywords

digital forensics, internet of things, iot, security

1. INTRODUCTION

Internet of Things (*IoT*) could be naively described as every object or embedded system that collects data and communicates with each other without human intervention. Origins of the term "Internet of Things" can be traced back to the 1985, where Peter T. Lewis coined it in speech he delivered at a U.S. Federal Communications Commission. *IoT* is an important topic in technology and engineering industry as it embodies a wide spectrum of networked products, systems and sensors. Although it strives to enhance almost every aspect of our lives, there are its disadvantages, namely non-regulated, non-standardized development and lack of security infrastructure. Aforementioned issues might have made such systems vulnerable to security exploits and make life difficult for forensic investigators that try to piece together evidence from enormous amounts of data to positively identify a suspect.

Why should field of digital forensics care about *IoT*? At the moment, there are more connected devices than there are people and it is predicted that by 2020 each person is likely to have an average of 5 connected devices. By 2020 almost quarter of all enterprise security breaches and attacks will involve some form of *IoT*. In the same time about 10 percent of all Information Technology (IT) security budget will be allocated to *IoT* security and half of all *IoT* implementations will involve some form of Cloud security or service.

*Overview of presentation: Jessica Hyde, "IoT 4n6: The Growing Impact of IoT on Digital Forensics" (DFRWS USA 2018)

2. IoT DIGITAL FORENSICS

When we mention *IoT*, we are in most situations referring to the consumer scope of devices, which comprise of wearables, home automation (smart home appliances) and "*Things that Go*", latter includes smart cars, car automation, drones, etc [4].

Majority of users sees main benefit of *IoT* in form of security, because it gives them more information about themselves or their property. But with issues in security infrastructure, that would hardly be the case. Main reason for use of *IoT* and related products is mainly convenience. It helps us with automation of everyday tasks, so that we can focus on other things. Another scope of *IoT* implementations, that is quickly becoming widespread, involves enterprise and corporate environments. For example, logistical companies for tracking and monitoring shipments, factories for machine monitoring, etc.

2.1 IoT device types

Previously we mentioned different types of *IoT* devices. Regarding digital forensics those can be roughly split into three groups, by what kind of data we can extract from them [4].

2.1.1 Wearables (fitness trackers, smart watches, etc.)

Group includes devices that are mostly worn, like fitness bands/trackers or smart watches. While it tends to refer to the items that can be put on and taken off, there are more invasive versions in case of implanted devices such as micro-chips.

From these devices we can get hardly any data directly from its hardware because of its small size and even smaller components. Most of the data collected is transferred and can be accessed from paired device or some other storage medium (in many cases cloud service). Since smart watches act more than just normal watches and are similar to fitness bands, most data acquired is similar.

Collected data includes information about individuals' activity, time frames and level of it. Analysed daily activities useful for determining life patterns. Since most of them include heart sensors, it can be informational, if the data is graphed and interesting anomalous data examined.

2.1.2 Home automation (smart home appliances)

These devices are usually larger than wearables, for example smart speakers, refrigerators, smart TVs, etc. One of their

benefits is their customizability, through which they can be tailored to the users' demands. Although most modern appliances have programs that follow distinct set of steps, home automation brings it to the next step in form of centralized control.

Devices feature some sort of onboard storage, some even in form of storage cards, but it is mostly not easily accessible without chip-off procedures. Fact that the documentations for chip locations and board schematic are non-existent, makes this even harder. These procedures are highly invasive techniques, that could irreversibly damage device circuitry, but may be the only option to collect device data.

We still get most of the data from paired devices or other storage. For example smart speakers can cache an audio record of last request (that could last up to 60 seconds), so it is imperative we do not use "wake" words when conducting an investigation where these types of devices are present, since we could overwrite the cache. Those audio records can be found in cloud storage (in case of *Google Echo*), but jurisdictions and permissions might apply. Cloud data records include requests (their "punch" words), audio recordings and timestamps. Security issues apply regarding application stores for such devices, which are mostly user developed applications, free, uncurated and could pose as another attack vector.

2.1.3 "Things that Go" (car automation, drones, etc.)

Devices that fall into this category are among the largest form of automation, namely self-driving cars, drones, car automation. Here the focus will be on car automation that includes in-vehicle security, hands-free telephony, navigation and diagnostics. Car manufacturers are developing increasingly sophisticated systems built on components like sensors, application programming interfaces and micro-services.

Hardware in most cases include full onboard computer with some form of proprietary storage system and protocol. It also includes some kind of communications device (mobile network) for uploading and downloading data (routes, traffic info, etc.) Data can be accessed using specialised tools and devices.

Although you cannot get navigation data from paired device (usually phone), useful data can still be recovered. You can retrieve points of interest from navigation, if there was GPS navigation request, start and end points can be recovered (although there is no information if the route was actually driven, but could provide some locational information). If car has a wireless carrier (SIM card), you could retrieve cell tower information, which provides quite good estimation about geo-location.

2.2 Traditional versus IoT digital forensics

Digital forensics could be described as a process that is used to identify the digital evidence in its most original form and then performing a structured investigation to collect, examine and analyse the digital evidence. There are number of factors that should be considered in IoT crime scene. One of such is kind of hardware involved. IoT is envisioned as

a system that involves communication between wide variety of devices. That introduces a new dimension to digital forensics in terms of what systems are seized or cordoned off for an investigations. Section describes few of the larger dimensions that IoT will introduce or affect in regard to usual digital forensics [1].

2.2.1 Evidence sources

While in traditional digital forensics evidence could be some kind of computer system, mobile device, storage device, network device in IoT digital forensics these things can be home appliances, cars, sensor nodes, medical implants (human or animal). This disparity of device types will introduce interesting challenges for device-level investigators.

2.2.2 Data types

In terms of evidence data types, IoT could be any possible format, it could be a specific format for a particular vendor. Comparing it to the traditional digital forensics where data could be in some form of electronic record or standard file format. IoT data would have to be unravelled and placed in understandable and usable format.

2.2.3 Network infrastructure

IoT limitations encourage vendors to develop new specialized protocols to fit this limitation. Because of this network lines become unclear comparing it to the traditional networks. Increasing in the blurry boundary lines makes seizing IoT devices one of the challenges of IoT digital forensics.

2.2.4 Jurisdiction and Ownership

Although we previously mentioned that IoT will change some aspects, this is one of the few that remain largely unchanged. Owners could be individuals, groups, companies, governments, etc. While device ownership is clear in most cases, data ownership sees the same jurisdiction and permission problem as in traditional digital forensics, especially regarding cloud stored data.

3. CHALLENGES OF IoT DIGITAL FORENSICS

The IoT digital forensics will undoubtedly increase the amount of evidence from physical world than conventional computer systems. Because without the need for user interaction, IoT environments are likely to contain contextual evidence of which the perpetrators are oblivious.

Four main phases of digital forensic face a number of challenges in IoT [3].

3.1 Identification

As mentioned in previous chapter, detecting presence of IoT in a crime scene poses a challenge to digital forensic investigations, as does identification of a particular users' data. This complicates "search & seizure" phase, because you need to have an IoT specialist present to identify possible data sources or destinations.

3.2 Preservation

Traditional digital forensics includes established protocols for capturing volatile evidence, for example memory dumps

before shutting down the machine. Evidence volatility is much more complex in *IoT*. Data may be stored locally by another device, in which case lifespan of the data before being overwritten, deleted, compressed is finite. Stored data could be transferred to the cloud and processed, aggregated or replaced by data from another device (ad-hoc networks).

3.3 Analysis

Analysis of *IoT* environment data will have to consider its origin to demonstrate the evidence is reliable and authentic. Comparing this to conventional digital forensic investigations where the main consideration is temporal, for example file created, accessed, modified and email time lines. Interaction between *IoT* and cloud services brings us that most data is aggregated and processed by those services and it is likely that *IoT* evidence will be the subject of cloud investigation.

3.4 Presentation

Presenting the findings of *IoT* investigations could pose a challenge since data might be processed and aggregated, what could alter the structure and meaning of data. Even the semantics of the data could pose problems with presenting the data, because same structured data (temperature, etc.) could have different meanings that could prove misleading.

Besides mentioned phases of digital forensic investigations, there are other aspects of *IoT* digital forensics that will bring challenges in its investigations [1][5].

3.5 Data location

Much of the user data is spread in different locations, which could be out of user control. Data could be located in the cloud (third party location), paired device (mobile phone), or device itself. Because of that one of the biggest challenges in *IoT* digital forensics is identifying the location of evidence. Storage of data in multiple locations with multiple jurisdictions is already an issue in conventional digital forensics due to difference in laws. *IoT* will add even more complexity as devices travel between different networks (private, personal and public)

3.6 Data format

In most cases device generated data format is not matching to what is saved in the cloud (or data on paired device). Besides that, users have no direct access to the data. Moreover as was already mentioned, data could be aggregated, processed or compressed, therefore for data to be accepted as evidence, it should be returned to original format prior to performing an analysis.

3.7 Data quantity

Because of sheer number of devices in interconnected networks that communicate with each other, we have an "explosion" of data. In addition to storing or processing exabytes (billion of gigabytes) of data the problem is sifting through all that data, which in turn will take excessive amount of time, to find reliable and relevant information.

3.8 Data lifespan

Storage limitation of *IoT* devices mean, that the lifespan of data is short and easily overwritten, which results in possible loss of relevant evidence data. This could be easily circumvented by transferring it to the local hub or cloud service, but that brings another challenge with securing the chain of evidence and how to prove evidence was not changed or modified in any way.

3.9 Network boundaries

Conventional digital forensics have clearly defined network based investigations and network boundaries. If some network device is located at the crime scene, we can in most cases assume that its network was the source or the destination. Comparing that to *IoT*, where area networks, merge with each other and *IoT* devices change multiple in any time frame. Challenge for *IoT* investigators is collecting relevant information from *IoT* device, that travelled between multiple networks, leaving multiple digital fingerprints. Consequently this might aid investigators by providing some form of *IoT* device traceability. Another challenge blurry network boundaries bring are privacy issues with collecting data in areas where personal data is being collected (hospitals). Regarding that, obtaining the right type of permissions to seize and analyse these should be a subject of discussion in *IoT* development.

3.10 Cloud anonymity

Most cloud services do not require relevant or accurate information for user to sign up. As a consequence, even if investigators can prove, that data in the cloud came from the *IoT* device at the crime scene, that does not mean the evidence will positively identify the criminal.

3.11 Security

Major security challenges in *IoT* environment arise, because of wide *IoT* node distribution and private nature of the data. This section will briefly describe few of the larger security challenges [2].

3.11.1 Authentication and authorization

Authentication includes authentication of peer-to-peer channel that is transferring data and authentication of the data source. With that we can ensure that chain of evidence and reliability and integrity. Absence of Certificate Authority (CA), we need other infrastructure for validating cryptographic keys. And because of limited *IoT* resources, secure key generation and exchange should not cause major overhead. Authorization on the other hand involves specification of access rights to different resources while access control guarantees access rights of only authorized resources. Each and every *IoT* node, might only support limited and different verification mechanisms, therefore deploying and management of authorisation mechanisms could prove problematic for heterogeneous *IoT* network.

3.11.2 Privacy

IoT devices that sense private information (health data) pose a significant threat to privacy unlike conventional systems, where user compromises his or her own privacy, *IoT* environments

are collecting information without them noticing and identifying such nodes that passively collect information is a huge challenge in heterogeneous *IoT* networks.

3.11.3 Development

Development of *IoT* devices is increasing over the last few years, which means better (and smaller) hardware is being developed at an increasing pace. Consequence of this development is, that old devices get discontinued, vendors stop software updates (firmware included) or drop support for the device altogether. All this means, we will see an increase of such devices, that could be vulnerable to attack, or used as an attack vector for compromising larger systems.

3.11.4 Infrastructure

Lastly, one of the more pressing issues is lack of security experts. Since *IoT* is quite "young", most companies and developers disregard security infrastructure of their devices. Substantial part of them lack proper security departments or do not have them at all, which means device security is inadequate, basic or omitted altogether.

4. CONCLUSION

Fast paced development of *IoT* infrastructure is followed by a variety of forensic and security challenges. New techniques compared to conventional digital forensics are required to overcome the challenges and leverage the architectures and processes to access this large-scale source of potential evidence.

Development of frameworks and guidelines for *IoT* investigators should be a major consideration in advancement of *IoT* digital forensics. Further work could improve gathering of data from device hardware (JTAG/chip-off). Cloud investigations bring a variety of legal challenges that should be addressed or at least taken into consideration with development of *IoT*. Encryption of mobile devices should be considered, since it could block a substantial portion of *IoT* evidence.

Lastly, many aspects of *IoT* infrastructure are non-standardized, undocumented and unregulated, which should be one of the main topics of discussion in development and upgrades of *IoT* infrastructure.

5. REFERENCES

- [1] S. Alabdulsalam, K. Schaefer, T. Kechadi, and N.-A. Le-Khac. Internet of things forensics—challenges and a case study. In *IFIP International Conference on Digital Forensics*, pages 35–48. Springer, 2018.
- [2] M. Conti, A. Dehghanianha, K. Franke, and S. Watson. Internet of things security and forensics: Challenges and opportunities, 2018.
- [3] R. Hegarty, D. J. Lamb, and A. Attwood. Digital evidence challenges in the internet of things. In *INC*, pages 163–172, 2014.
- [4] J. Hyde. Iot 4n6: The growing impact of iot on digital forensics, 2018.
- [5] E. Oriwoh, D. Jazani, G. Epiphaniou, and P. Sant. Internet of things forensics: Challenges and approaches. In *9th IEEE International Conference on Collaborative computing: networking, Applications and Worksharing*, pages 608–615. IEEE, 2013.



7 DISKOVNA FORENZIKA DISC FORENSIC

Analysis of the BTRFS File System Using a Storage Pool Extension of The Sleuth Kit

Sandi Režonja

Faculty of Computer and Information Science
University of Ljubljana
sr3182@student.uni-lj.si

Gregor Robert Krmelj

Faculty of Computer and Information Science
University of Ljubljana
academic@krmelj.xyz

ABSTRACT

BTRFS is a Linux based file system that almost exclusively uses B-trees to store its structure according to the copy on write principle. The file system enables us to use subvolumes, which can be viewed as mountable directories in order to organize our directory structure and roll back to previous snapshots of the file system. Unlike older file systems, BTRFS was designed for pooled storage.

The Sleuth Kit is a forensic toolkit that allows forensic analysis regardless of the file system. Hilgert et al. implemented an extension for TSK with BTRFS support. Their implementation provides all essential commands to perform forensic analysis of the BTRFS file system even when disks are missing from the pool.

We attempt to replicate Hilgert et al. results of JPG image data extraction using their BTRFS extension of The Sleuth Kit while one out of two BTRFS disks are missing. We are able to partially recover text files, but images are either unreadable or showed no discernible content.

Keywords

Digital forensics, The Sleuth Kit, BTRFS, pooled storage

1. INTRODUCTION

BTRFS (B-Tree file system) is a Linux file system whose development started in 2007, making it a younger file system compared to other common Linux file systems such as the ext* family [11]. The primary goals of BTRFS are to deal with the lack of pooling, snapshots, checksums and integral multi-device spanning in older Linux file systems [1]. Despite its young age, BTRFS has played a role in increasing efficiency and resource utilization of Facebook's data centers [8].

The Sleuth Kit (TSK) is a forensic toolkit that works independently of the file system [12]. Hilgert et al.[10] expanded

TSK to support BTRFS for forensic analysis. The main challenge is the spanning of a single file system through multiple storage volumes whereas other (older) file systems have a single volume. To solve this an additional step is added to the forensic analysis model - pool analysis.

At first, we give an overview of the BTRFS file system and its main characteristics (significant for forensic analysis). Then we present a relevant summary of how Hilgert et al.[10] integrated BTRFS into TSK. We heavily rely on Hilgert et al. article for our findings. Lastly, we try to test the extended TSK on our own.

2. BTRFS FILE SYSTEM

BTRFS achieves its goals by using a special B-tree which is optimized for copy-on-write (CoW) [11]. By using CoW the file system does not modify data directly but rather on a copy. Such a technique makes it possible to support snapshots and having the file system in an always consistent state [3].

2.1 B-tree

A B-tree is a self-balancing tree data structure. A modified B-tree structure (figure: 1) optimized for use with the CoW principle and references that was proposed by IBM researcher Ohad Rodeh is used in BTRFS almost exclusively [11]. This includes: storing data, metadata, space allocation records and the trees themselves. Even more so, the same code is used to implement all the aforementioned data, this way a single algorithm is needed for searching and maintaining essential file system structures [3].

Unlike binary trees, each B-tree node contains multiple elements and child nodes. Every time the number of elements falls over or under the set parameters the tree is rebalanced. In BTRFS interior nodes contain key-pointer pairs that point to child nodes and leaf nodes contain key-data pairs [1].

Essential B-trees used in BTRFS are [2]:

- **Chunk tree** - used for mapping logical addresses to physical.
- **Root tree** - contains logical addresses of all other trees.

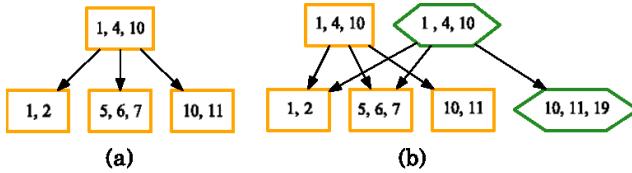


Figure 1: Example B-tree a) and the copied B-tree b) after adding element 19 in the BTRFS file system [11].

- **File system tree** - contains file and directory hierarchy, as well as metadata and references to data for each subvolume/snapshot.
- **Extent tree** - stores allocation records for byte ranges that are in use. Extent items contain reference counts as well as back-references to tree nodes.
- **Checksum tree** - stores a 4-byte checksum for every 4k block of data.
- **Device tree** - the inverse of the chunk tree, used for mapping physical addresses to logical.

2.2 Subvolumes

Subvolumes can be thought of as mountable directories. It is common for file systems to have a single mountable root, in contrast, BTRFS offers the ability to mount each subvolume separately [4]. As expected, by mounting the default *top-level subvolume* we can see the entire file system including child subvolumes (which will appear as directories) [4]. Mounting specific subvolumes causes us to view the file system from the subvolumes perspective. The use of subvolumes is important since they are closely related to snapshots.

2.3 Snapshots

Snapshots are subvolume copies created based on the CoW principle. Until modified, a snapshot shares its data and metadata with the original subvolume [4]. Because a snapshot is under the hood a subvolume, a rollback is performed by mounting the snapshot at the desired mount point. Snapshots are not recursive and any nested subvolumes are only referenced within the snapshot [4]. The send-receive BTRFS utility can generate a binary `diff` between two snapshots that can be used for a simple form of master-slave replication or incremental backups [7].

2.4 Superblock

The superblock is a block on the disk whose location is fixed [3]. All vital data on other file system trees are located here with which the system can bootstrap. To improve resilience the superblock is stored on each device at fixed offsets of 64 KiB, 65 MiB, 256 GiB, 1 TiB and 1 PiB [3].

2.5 Data extraction procedure

By accessing the superblock we can figure out where the chunk tree is located and consequently access the root tree.

Using the root tree we can build the right file system tree and search for the corresponding inode which contains metadata and the ID to the extent [10]. Data can be extracted by mapping the extent to the physical addresses using the chunk tree.

2.6 Multiple devices

Storage space on multiple devices is shared by mapping logical addresses to the corresponding combination of devices and physical addresses [10]. The address space is split into chunks consisting of multiple devices and referenced by chunk items in the chunk tree [10]. Data in a chunk is striped across devices according to the selected RAID configuration. Metadata is by default stored in two copies, even if there is only one device present [5].

3. INTEGRATION OF BTRFS INTO THE SLEUTH KIT

The Sleuth Kit (TSK) is an open source toolkit that enables forensic analysis independent of the used file system [12]. It relies on the universal model for file system forensic analysis introduced by Brian Carrier. Carrier's model divides file system forensic analysis into four stages [6]:

1. **Physical media analysis** - data is acquired from devices.
2. **Volume analysis** - acquired data is scanned for volumes (partitions, RAIDs, and logical volumes of volume groups)
3. **File system analysis** - volumes are interpreted as file systems and directories, files and metadata is extracted.
4. **Application analysis** - collected data is interpreted for its content.

As mentioned, newer file systems like ZFS and BTRFS associate multiple volumes with a single file system [9]. Hilgert et al. expanded this model (figure: 2) for analyzing pooled file systems by introducing *pool analysis* as an additional step in between volume analysis and file system analysis [10]. In pool analysis, volumes are grouped to form a pool for the corresponding file system as well as functionality to access file system data and metadata by calculating correct members and offsets [9].

Hilgert's et. al. expanded model is used to integrate BTRFS into TSK. BTRFS stores a superblock at the physical offset 0x10000 on every device [10]. The superblock contains a file system UUID, which is used to combine the right members into a pool. Missing devices can be determined by examining device ID's in chunk items from the superblock or from the chunk tree if all devices storing the chunk tree are available [10]. A new `p1s` command performs and displays results of the pool analysis [10]. The mapping of logical addresses to physical is done using the chunk tree. The correct stripe¹, stripe unit and offset within a stripe unit are calculated. See

¹The term stripe comes from RAID (striping) where it means a unit of data when striping. Striping is the process of segmenting data across multiple disks.

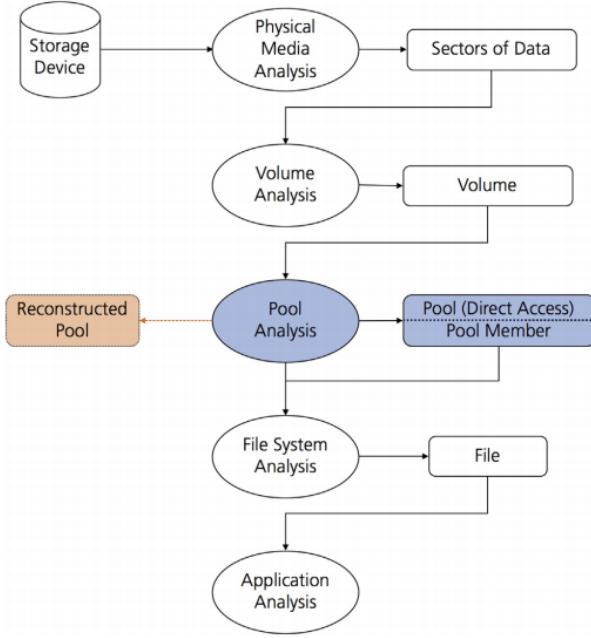


Figure 2: Hilgert et al. expanded model for pooled file system forensics analysis [9].

figure 3 for a visualization of a stripe configuration. Other important TSK commands that were expanded for BTRFS are [10]:

- **fsstat** - returns general file system information including snapshots and subvolumes.
- **fls** - lists all files and directories, including their ID's. This command can be applied to the file system itself or a subvolume including snapshots.
- **istat** - displays metadata for a given object ID.
- **icat** - extracts data associated with a metadata structure.

With each transaction, a new root tree is generated based on the CoW principle [10]. BTRFS stores four older versions of root trees that can be accessed using the **pls** command. This allows the recovery of some deleted files.

BTRFS does not allow you to mount the file system with a missing device [10]. This limitation was one of Hilgert's et al. main goals while extending TSK. Their contribution allows us to view present data by simply replacing missing data with zeros.

4. EXPERIMENTAL WORK

Hilgert et al. demonstrated analyzing a multi-device spanning BTRFS file system which contained a JPG image while one or two disks were missing out of three. They successfully extracted the image, although it had missing horizontal stripes of data, the image contents were discernible.

We attempted to replicate their results in a scenario where one of two disks is missing. We set up a BTRFS file system spanning two disks in a RAID0 configuration for data and RAID1 configuration for metadata on a virtual machine. We placed six images of different sizes and formats inside our newly created BTRFS file system along with two text files - one stored as ASCII text, the other UTF8. Afterward, we used Guymager, an open-source forensic imager, in order to create a *dd* image of the two disks. On our forensic virtual machine, we cloned Hilgert's et al. fork of TSK [14] and built the software.

At first, we were unable to run the *pls* command (in fact any pool aware TSK command) without getting a segmentation fault. After contacting Hilgert we came to the realization that their implementation does not accept full disk images but expects a raw BTRFS partition image and directory path of all pool aware extended commands expect a directory where the contents are just the raw BTRFS partition images. This design does not follow the TSK established workflow, which could be the reason their pull request [13] has not yet been merged upstream.

4.1 Recovering Files

First, we used TSK's *mmls* command to gather information where the BTRFS partitions start and stop. By using *dd* we created a raw copy of the BTRFS partitions. We placed each raw partition image inside its own directory. Using the command *fls* with the pool flag *-P* and the directory of the first partition image (figure: 7), we were able to get a list of all the files on the file system. This is possible using only one disk in RAID0 because during setup we set up the metadata to be stored in RAID1 (default file system setup).

The *fls* command provided us with the inode of each file on the filesystem. Using this information along with the TSK command *icat* in pool mode *-P* we were able to recover all files from the file system (figure: 4).

We then tried to see if the contents of the files matched the original and if we could recover any content. For all image files we were unable to recover the image to the point where Hilgert, et.al. were, even using multiple (free) image recovery software. JPG files were unreadable, while PNG images displayed no relevant content (e.g. figure 5).

However, we were able to partially recover both of the text files (encoded as ANSI and UTF-8). Both files contained NUL characters where data was missing, as shown in figure 6. One file starts with NUL characters and the other contained them later on.

5. CONCLUSION

BTRFS is a young file system with modern features such as pooled storage and snapshot support. Hilgert et al. created a fork of The Sleuth Kit which allows forensic analysis of devices that are a part of a BTRFS file system. However in our attempt to replicate a similar case as Hilgert et al., we came to the conclusion that the tools available are not well tested and unfit for a forensic analysis of BTRFS file systems.

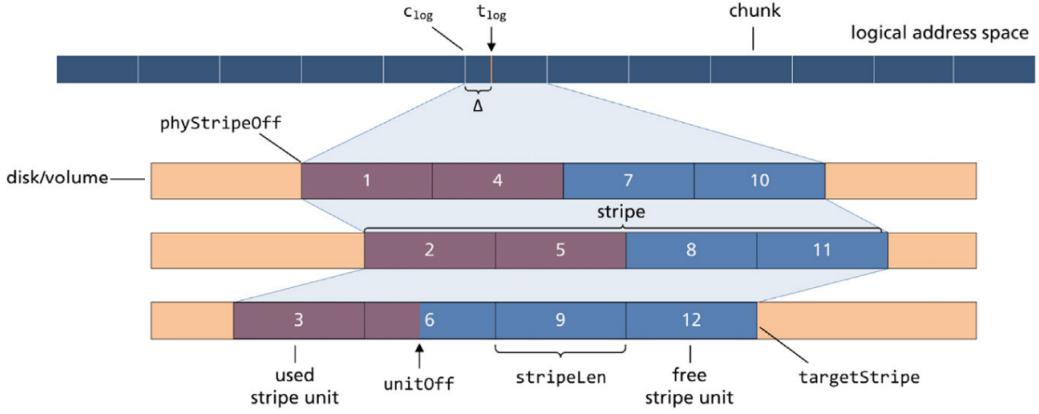


Figure 3: Stripe structure in a RAID0 chunk [10]. Marked in the figure are values used for calculating the physical location of data at the logical address t_{log} .

```
icat -P ./first_disk/ 257 > sample_image_10mb.jpg
```

Figure 4: Using *icat* to recover files from the BTRFS file system



Figure 5: Example of original and recovered *sample_images_500kb.png* image.

We hope that Hilgert et al. will try to improve their work integrating BTRFS support into TSK and pushing it upstream for it to be included in the official toolkit. As Hilgert et al. have noticed a pull request since March 27, 2018, on the official TSK repository that claims to offer support for single image BTRFS file systems has until the time of writing still not merged into the official branch nor any other visible activity.

There is a lot of room for future work in this area, especially improving current attempts to expand forensic tools with BTRFS support. With companies such as Facebook adopting BTRFS as their file system we predict an increase in situations where digital forensic professionals will come across BTRFS. With the lack of tools in this area, proper forensic analysis will be difficult and time consuming.

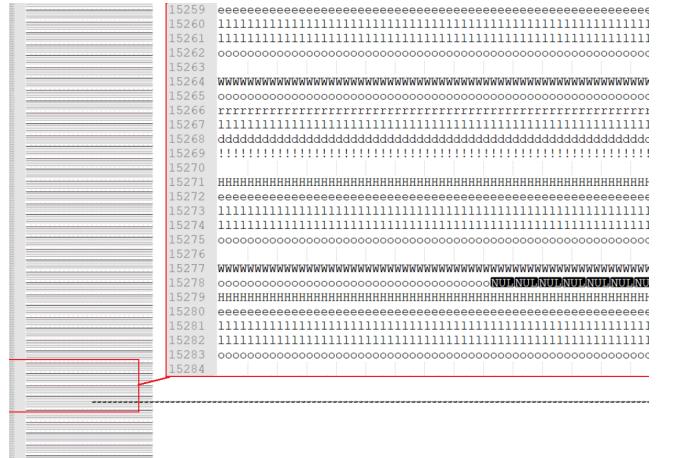


Figure 6: Contents of a recovered text file.

```
$ fls -P first_disk/
r/r 257:      sample_image_10mb.jpg
r/r 258:      sample_image_1mb.jpg
r/r 259:      sample_image_30mb.jpg
r/r 260:      testfileANSI.txt
r/r 261:      testfileUTF8.txt
r/r 262:      sample_image_1mb.png
r/r 263:      sample_image_500kb.png
r/r 264:      sample_image_5mb.png
```

Figure 7: *fls* command output when recovering files from a single disk

6. REFERENCES

- [1] Btrfs. <https://en.wikipedia.org/wiki/Btrfs>, 2019.
Last accessed: 29 April, 2019.
- [2] Btrfs wiki: Btrees.
<https://btrfs.wiki.kernel.org/index.php/Btrees>,
2017. Last accessed: 29 April, 2019.
- [3] Btrfs wiki: Glossary. <https://btrfs.wiki.kernel.org/index.php/Glossary>,
2017. Last accessed: 29 April, 2019.
- [4] Btrfs wiki: Sysadminguide. <https://btrfs.wiki.kernel.org/index.php/SysadminGuide>, 2017. Last accessed: 29 April, 2019.
- [5] Btrfs wiki: Using btrfs with multiple devices.
https://btrfs.wiki.kernel.org/index.php/Using_Btrfs_with_Multiple_Devices, 2017. Last accessed: 29 April, 2019.
- [6] B. Carrier. *File System Forensic Analysis*. Addison-Wesley Professional, 2005.
- [7] J. Corbet. Btrfs send/receive.
<https://lwn.net/Articles/506244/>, July 2012. Last accessed: 4 May, 2019.
- [8] Facebook open-sources new suite of linux kernel components and tools.
<https://code.fb.com/open-source/linux/>, 2018.
Last accessed: 29 April, 2019.
- [9] J.-N. Hilgert, M. Lambertz, and D. Plohmann.
Extending the sleuth kit and its underlying model for pooled storage file system forensic analysis. *Digital Investigation*, 22:S76 – S85, 2017.
- [10] J.-N. Hilgert, M. Lambertz, and S. Yang. Forensic analysis of multiple device btrfs configurations using the sleuth kit. *Digital Investigation*, 26:S21 – S29, 2018.
- [11] O. Rodeh, J. Bacik, and C. Mason. Btrfs: The linux b-tree filesystem. *Trans. Storage*, 9(3):9:1–9:32, Aug. 2013.
- [12] Sleuth kit. <https://www.sleuthkit.org/sleuthkit/>, 2019. Last accessed: 4 May, 2019.
- [13] The sleuth kit btrfs support pull request. <https://github.com/sleuthkit/sleuthkit/pull/413>. Last accessed: 4 May, 2019.
- [14] The sleuth kit.
<https://github.com/fkie-cad/sleuthkit>. Last accessed: 4 May, 2019.

Uporaba časovnega žiga v ext4 za skrivanje podatkov

Andrej Rus
27182075

Anže Dežman
63140041

Domen Balantič
27172041

POVZETEK

Ext4 je priljubljen datotečni sistem, ki ga uporablja Android in številne distribucije Linuxa. Ta dokument analizira izvedljivost uporabe časovnih žigov datotečnega sistema ext4 za skrivanje podatkov. Najprej preučimo uporabo, strukturo in zmogljivost razpoložljivih časovnih žigov. Rezultati razkrijejo, da je del nanosekundnih časovnih žigov ext4 mogoče uporabiti za izgradnjo steganografskega sistema. Poleg tega opisemo ext4 anti-forenzično tehniko, ki omogoča tajnost skritih podatkov in enostavno uporabo v širokem spektru scenarijev. Opisan je niz zahtev (npr. nerazpoznavnost rednih in nepooblaščenih časovnih žigov) in koncept, ki lahko prikrije poljubne podatke v časovnih žigih datotečnega sistema.

Ključne besede

Digitalna forenzika, skrivanje podatkov, ext4, nanosekundni časovni žigi, steganografija, anti-forenzička, forenzička datotečnih sistemov

1. UVOD

Zaradi porasta kibernetskega kriminala in povečevanja anti-forenzičnih orodij bi bila dobrodošla obsežnejša analiza trenutno znanih anti-forenzičnih groženj, z namenom, da bi lahko tekom digitalne forenzične preiskave zbrali zanesljive dokaze in obenem razvili orodja za zaznavanje anti-forenzičnih tehnik. Vendar pa je skupno število znanstvenih člankov, osredotočenih na anti-forenzične študije, omejeno.

K. Conlan je s skupino raziskovalcev leta 2016 objavil analizo znanih anti-forenzičnih orodij, ki vključuje podatke o 308 orodjih in jih klasificira v posamezne kategorije [3]. Večina orodij spada v kategorijo skrivanja podatkov, kar je posledica vse večje potrebe po varovanju izmenjave in shranjevanja informacij. Skrivanje podatkov je možno na različnih nivojih, npr. na samem trdem disku, znotraj datotečnega sistema - znotraj rezerviranega prostora, znotraj "slack" prostora ali z zlorabo obstoječih datotečnih struktur.

Ext4 je priljubljen datotečni sistem, ki ga uporablja Android in številne distribucije Linuxa. Za samo prikrivanje podatkov se lahko uporabijo anti-forenzične tehnike, kot je to npr. skrivanje podatkov. V nadaljevanju se bomo osredotočili na potencialnega skrivališča podatkov znotraj ext4 datotečnega sistema. V času forenzične preiskave je pomembno vedeti za takšne prostore, zlasti zaradi vse obsežnejše uporabe ext4 datotečnega sistema.

Osredotočili se bomo na pristop, ki za skrivanje uporablja podatkovne strukture shranjene v inod tabeli. Inod tabela vsebuje vse metapodatke o datoteki ali datotečnem imenu (tj. podatke o lastniku datoteke, dovoljenjih datoteke, časovnih žigih s podatki o tem kdaj je bila datoteka ustvarjena, nazadnje spremenjena in izbrisana). Skritih podatkov, ki se ujemajo z običajnimi inod strukturami, orodja za analizo ne bodo prepoznala, saj pri shranjevanju oz. spremjanju teh podatkov ni zagotovljene doslednosti in zato nič nepričakovane, če se npr. časovne žige s skritimi podatki interpretira kot navadne časovne oznake. Brez dodatnih opozoril obstoječih forenzičnih orodij ali podrobnejšega preverjanja datotečnega sistema je takšne skrite podatke izjemno težko najti.

Namesto šifriranja zaupnih podatkov z dobro znanimi kriptografskimi tehnikami bomo obravnavali steganografski pristop, saj proces steganografije ne pušča nobenega dodatnega dokaza o izmenjavi informacij in zato ne vzbuja suma forenzičnega preiskovalca. Če bi namesto steganografije uporabljali kriptografijo, bi lahko tretje osebe prepoznale šifrirano komunikacijo, vendar ne bi znali razbrati njene vsebine.

2. SORODNA PODROČJA

2.1 Anti-forenzička

Izraz anti-forenzička je na podlagi preteklih definicij K. Conlan leta 2016 opredelil kot "kakršne koli poskuse, z namenom spremjanja, prekinitev, zanikanja ali kakršnega koli poseganja v znanstveno veljavne postopke forenzične preiskave" [3]. Na podlagi izvirne, splošno sprejete anti-forenzične taksonomije, ki sta jo predlagala Rogers (2005) [8] in Conlan (2016) [3] je bila oblikovana celovitejša taksonomija, ki anti-forenzične tehnike kategorizira v več kategorij. Razširjena taksonomija vključuje pet kategorij: (i) skrivanje podatkov, (ii) brisanje artefaktov, (iii) zakrivanje sledi, (iv) napadi na forenzična orodja in forenzične procese, (v) zaznavanje anti-forenzičnih dejavnosti. Vsaka izmed kategorij vsebuje še dodatne podkategorije.

Metodo skrivanja, ki jo bomo obravnavali v nadaljevanju, je mogoče klasificirati v kategorijo (i) skrivanje podatkov razširjene taksonomije. Zlasti se ujema z dvema podskupinama kategorije: manipulacija datotečnega sistema in steganografski datotečni sistem.

2.2 Skrivanje podatkov znotraj datotečnega sistema ext

Skrivanje podatkov v metapodatkih datotečnega sistema so kot prvi opravili znanstveniki pod vodstvom R. Andersona leta 1998, ki so kasneje poskrbeli za razvoj steganografskega datotečnega sistema StegFS [2]. StegFS temelji na ext2 datotečnem sistemu in ljudem omogoča, da zanikajo obstoj skritih podatkov. Leta 2012 je K. D. Fairbanks omenil več potencialnih skrivališč v ext4 datotečnem sistemu, ki pa jih ni podrobnejše preučil [4]. Göbel in Baier sta leta 2018 v članku "Anti-Forensic Capacity and Detection Rating of Hidden Data in the ext4 Filesystem" predstavila, analizirala in ovrednotila različne tehnike skrivanja podatkov v datotečnem sistemu ext4, vendar se nista osredotočila na časovne žige ext4 datotečnega sistema [5].

Uporabni viri za analizo datotečnega sistema ext4 zaobjema sistemsko analizo datotečnega sistema (D. J. Wong 2016 [9]), opis novih funkcionalnosti datotečnega sistema ext4, kot so npr. nanosekundni časovni žigi (A. Matruh 2007 [6]) in izvorno kodo jedra operacijskega sistema Linux.

2.3 Steganografija časovnih žigov

S. Neuner je s skupino raziskovalcev leta 2016 predlagal uporabo časovnih žigov datotečnega sistema kot steganografskega kanala za skrivanje podatkov [7]. Na primeru uporabe NTFS datotečnega sistema so prikazali, da je podatke na ta način zares mogoče skrivati. Na podlagi osrednje ideje te raziskave bomo v nadaljevanju analizirali ali je znotraj ext4 datotečnega sistema možno skrивati na podoben način. Analizirali bomo korelacijo med entropijo pravilnih in ponarejenih časovnih žigov ter zmožnosti steganografskega kanala tj. koliko podatkov je mogoče skriti z uporabo predlagane tehnike in kakšna velikost datotečnega sistema je najprimernejša.

3. ANALIZA EXT4 ČASOVNIH ŽIGOV

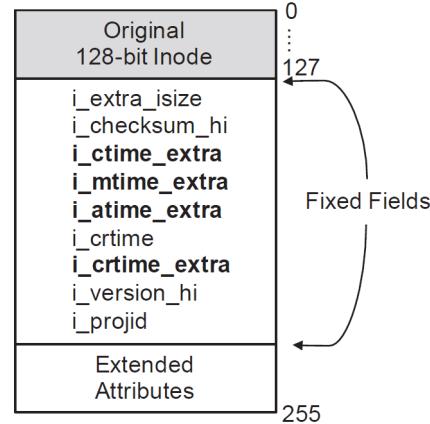
V tem razdelku analiziramo časovne žige ext4 s ciljem njihove uporabe kot steganografskega kanala. Najprej pregledamo pet razpoložljivih časovnih žigov v datotečnem sistemu ext4 in nato analiziramo, kateri časovni žigi in kateri del njih je koristen za naš pristop. Rezultat tega razdelka je, da so nanosekundni deli zadnjega dostopa do datoteke in časa kreiranja primerni za naš steganografski kanal.

3.1 Časovni žigi v ext4

Za izbiro primernih časovnih žigov za skrivanje podatkov najprej pregledamo vse razpoložljive časovne žige v ext4. Struktura inoda v ext4 je prikazana na sliki 1. Prvih 128 bitov ext4 inoda je enakih inodom iz preteklih verzij ext (t.j. ext2, ext3), ki jim sledijo nekatera dodatna fiksna polja, ki se začnejo na bitu z odmikom 128.

V nadaljevanju bomo pojasnili časovne označke, ki jih zagotavlja ext4 za vsako datoteko oziroma direktorij. Uporabljamo izraz objekt, ki označuje datoteko ali direktorij. Tabela 1

Ext4 Large Inode



Slika 1: Postavitev strukture inoda ext4.

prikazuje vse časovne žige ext4 z ustreznimi odmiki in dolžino znotraj inoda. Za vsak objekt ext4 zagotavlja naslednje časovne žige:

- čas zadnje spremembe objekta (mtime);
- čas zadnjega dostopa do objekta (atime);
- čas zadnje spremembe metapodatkov (ctime), npr. sprememba lastništva, dovoljenj ali velikosti datoteke;
- čas izbrisca (ctime)

Vsek od teh štirih časovnih žigov je shranjen v 32-bitni celem številu in predstavlja sekunde od začetka Unixa (1. januar 1970). V ext4 je bil dodan še peti časovni žig: čas kreiranja objekta (crtime).

Odmik	Dolžina	Ime	Opis
0 x 8	32 bits	i_atime	Čas dostopa
0 x C	32 bits	i_ctime	Čas spremembe metapodatkov
0 x 10	32 bits	i_mtime	Čas spremembe
0 x 14	32 bits	i_dtime	Čas izbrisca
0 x 84	32 bits	i_ctime_extra	Dodatni biti za ctime
0 x 88	32 bits	i_mtime_extra	Dodatni biti za mtime
0 x 8C	32 bits	i_atime_extra	Dodatni biti za atime
0 x 90	32 bits	i_crtime	Čas kreiranja
0 x 94	32 bits	i_crtime_extra	Dodatni biti za crtme

Tabela 1: Možni časovni žigi v strukturi ext4 inoda.

Poleg tega večja velikost strukture inoda (256 bajtov) v ext4 ponuja dodaten prostor za podporo nanosekundnih časovnih žigov. Štiri od petih časovnih žigov (razen žiga za čas izbrisca) so bili razširjeni na 64 bitov z dodajanjem 32-bitnega polja i_[c|m|a|cr]_time_extra, kot je prikazano na sliki 1. Spodnja dva bita teh štirih dodatnih 32-bitnih polj sta uporabljeni, da razširita 32-bitno sekundno polje v 34-bitno sekundno polje in tako preprečita overflow časa v letu 2038 (zdaj je novi datum overflowa 10. 5. 2446). Zgornjih 30 bitov dodatnega polja se uporablja za zagotavljanje nanosekund, saj 30 bitov zadostuje za podporo časovnim žigom z natančnostjo ene nanosekunde. Ext4 lahko zapolni dodatna

polja časovnih žigov z nanosekundno natančnostjo informacije, saj sistemski ura Linuxa ne omogoča takšne natančnosti.

A vendar, končni uporabniki, ki dostopajo do datotečnega sistema s pomočjo navadnega vmesnika operacijskega sistema lahko vidijo le časovne žige s sekundno natančnostjo. Tu gre za informacijsko vrzel med tem, kako moderni datotečni sistemi shranjujejo časovne žige in kako jih končni uporabniki uporabljajo. Običajni Linux raziskovalci datotek (*file explorers*), kot tudi ukaz ls -la, ne podpirajo nanosekundne natančnosti. Razen za v prihodnje ali pri časovno kritičnih aplikacijah, v večini primerov dodatna natančnost časovnega žiga ni potrebna. Tako manipulacija nanosekundnega dela časovnega žiga ext4 za končne uporabnike ni pomembna.

3.2 Analiza nanosekundnih časovnih žigov

Zgoraj navedena štiri dodatna polja časovnih žigov omogočajo skritje 16 bajtov podatkov pri vsakem vnosu v inod tabelo. Vendar, če bomo za skrivanje informacij uporabili spodnja dva bita, kar vodi do datumov po letu 2038, izgleda sumljivo in pomaga forenzikom razkriti skrite podatke in steganografski kanal. Slika 2 prikazuje, kako se uporabita spodnja dva bita za razširitev prvotne časovne oznake. Za dokazovanje tega vedenja je časovni žig dostopa namerno nastavljen na leto 2111. Ukazi Linuxa debugfs -R 'stat <inode>' '[image]' ali stat [file] lahko razčleni manipulirane časovne žige.

Seznam 1: debugfs

```
debugfs -R 'stat <12>' testimage.dd
ctime: 0x58aed1d5:90fbce3c -- Thu Feb 23 13:14:09 2017
atime: 0x0942d682:00000001 -- Sat Jan 10 15:15:30 2111
mtime: 0x70e87e82:00000000 -- Thu Jan 10 15:15:30 2030
crtim: 0x58aed1b7:03d70980 -- Thu Feb 23 13:12:39 2017
```

V seznamu 1 je prikazan izhod debugfs, seznama 2 pa prikazuje rezultat ukaza stat. Za razliko od debugfs, ukaz stat ne razčleni *crtim* in *dtim*. Da bi preverili razčlenitev časovnega žiga, dodamo spodnja dva bita polja *i_atime_extra* z vrednostjo 01 na začetek polja *i_atime*. Tako dobimo šestnajstico vrednost 0x010942D682 oziroma decimalno številko 4450342530. Pretvorba v človeku berljivo obliko se izvede z datumom -d @ 4450342530. Kot rezultat dobimo Sat Jan 10 15:15:30 2111, kar ustreza izhodom debugfs.

Da bi končno dobil del nanosekund (npr. časovnega žiga spremembe) moramo odstraniti spodnja dva bita, ker nista uporabljeni za štetje nanosekund, in zamakniti (*shift*) vrednost 0x90FBCE3C na desno za dva bita. To je enakovredna deljenju s 4. Kot rezultat te operacije dobimo 0x243EF38F ali 608105359, kot je prikazano v seznamu 2.

Seznam 2: stat

```
stat testfile.txt
Access: 2111-01-10 15:15:30.000000000 +0100
Modify: 2030-01-10 15:15:30.000000000 +0100
Change: 2017-02-23 13:13:09.608105359 +0100
```

3.3 Časovni žigi kot prenaložci steganografskih sporocil

Naslednje se bomo posvetili vprašanju, kateri časovni žigi ext4 so primerni za izgradnjo robustnega steganografskega sistema. Kot smo videli, je smiseln prikriti podatke le v zgornjih 30 bitih dodatnih polj časovnih žigov, ker je preostalih 34 bitov vidnih končnemu uporabniku. Nedoslednosti bi lahko povzročile sum med forenzično preiskavo. Odvisno od števila (neuporabljenih) časovnih žigov je razpoložljiv prostor za skrivanje v velikosti od 30 bitov, če je uporabljen samo en časovni žig, do 15 bajtov, če so uporabljeni vsi štirje časovni žigi znotraj tabele inod. Ker sodobni datotečni sistemi običajno vsebujejo tudi do milijon datotek, nam to skupaj predstavlja nekaj megabajtov, ki se lahko uporabijo za skrivanje tajnih podatkov.

Tako kot v drugih datotečnih sistemih, se tudi v ext4 časovni žigi lahko kadarkoli prepričejo zaradi interakcije uporabnika (npr. spremembe vsebine datoteke) ali interakcije sistema (npr. spremembe dovoljenj ali velikosti datoteke), ker so to časovno omejeni metapodatki, ki opisujejo le trenutno stanje datoteke ali imenika. Tako je spremenljivost časovnih žigov odvisna od scenarija uporabe. Nekateri časovni žigi se bodo sčasoma spremenili, ostali pa bodo ostali nespremenjeni. Časovni žig kreiranja datoteke je staticen in zato primeren za skrivanje podatkov, saj se nanaša na edinstven dogodek, ko je datoteka ustvarjena. Medtem ko se časovna žiga za spremembo metapodatkov ali spremembo datoteke posodobita vsakič, ko se spremeni vsebina datoteke. Taka posodobitev bi uničila skrite steganografske informacije, zato se običajno staticne informacije uporabljajo kot nosilec za steganografske sisteme. Kljub temu je pomembno omeniti, da jih je teoretično mogoče uporabiti, dokler ni niti sprememb vsebine datoteke, niti sprememb metapodatkov (npr. pri uporabi samo staticnih datotek, kot je JPEG datoteka, kjer ne spominjam atributov metapodatkov).

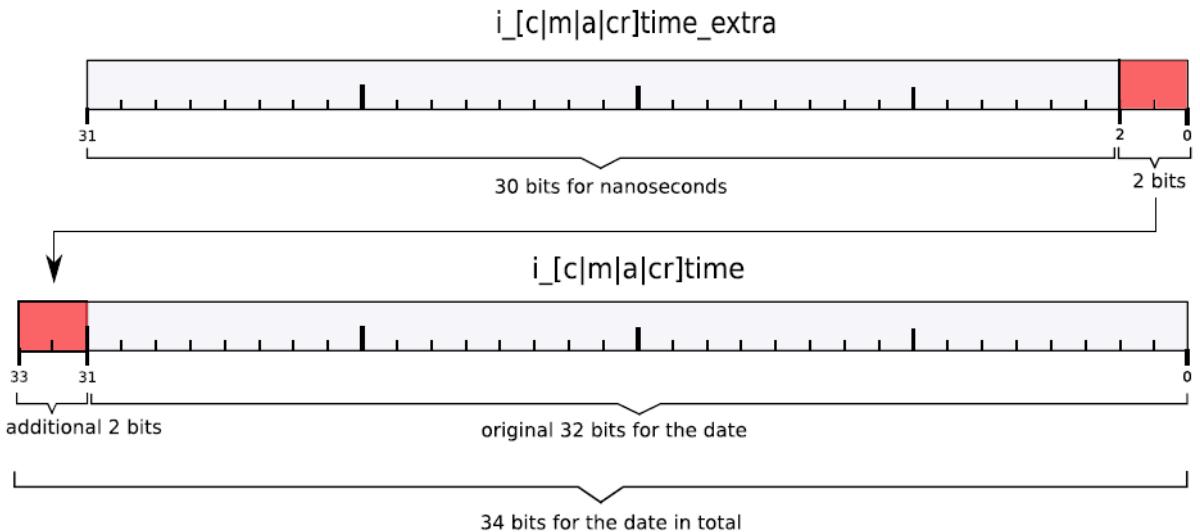
Časovni žig dostopa se posodobi ob vsakem dostopu do datoteke oziroma odprtju direktorija. A da bi zmanjšali obremenitev datotečnega sistema in povečali učinkovitost, lahko onemogočimo ponavljajoče se posodobitve časovnih žigov za dostop. To je privzeto v mnogih sodobnih datotečnih sistemih (npr. Microsoft uporablja to funkcijo od Viste naprej) in se pogosto priporoča za SSD diske in pogone USB. Na Linuxu se da nastaviti to možnost v /etc/fstab datoteki.

Ker želimo informacije trajno skriti, je zanimivo kaj se zgodi z vrednostmi časovnega žiga, ko je datoteka izbrisana. Medtem ko biti v *i_ctime_extra* in *i_mtime_extra* dobijo novo vrednost, ki ustreza času, ko je bila datoteka izbrisana, na *i_atime_extra* in *i_crtim_extra* operacija brisanja ne vpliva. Torej, če za steganografsko uporabo uporabljamo samo časovni žig dostopa in kreiranja, brisanje datoteke ne vodi neposredno do težav pri obnavljanju skritih podatkov. Kljub temu inod izbrisane datoteke postane neallociran in tako lahko prostor, kjer je bil, prosto uporabljamo. Lahko se zgodi, da je skrita vsebina prepisana in je postopek obnovitve odvisen od ustreznih metod popravljanja napak.

4. METODOLOGIJA

Namen predlagane anti-forenzične tehnike je prikriti podatke v časovnih žigih ext4, ki služijo kot steganografski nosilci podatkov.

Glavni cilj takšne tehnike je zagotavljanje dejstva, da napa-



Slika 2: Uporaba originalnega ext4 časovnega žiga v kombinaciji z dodatnimi 32-bitnimi polji.

dalec ne more odločiti, ali so znotraj časovnih žigov skriti kakšni podatki ali ne. S stališča končnega uporabnika bi si obenem želeli pripraviti takšno orodje, ki bo razumljivo tudi uporabniku z majhnim tehničnega znanja. Zato se bomo osredotočili steganografski sistem za shranjevanje z visoko stopnjo tajnosti in enostavno uporabo.

Tajnost pomeni nezmožnost izluščitve skritih podatkov znotraj časovnih žigov. Zato bomo uporabili dvojni pristop, ki temelji na steganografiji in kriptografiji. Vemo, da lahko s pomočjo steganografije zakrijemo obstoj shranjenih informacij znotraj časovnih žigov vse do trenutka, ko nekdo ponovno dostopa do izbrane datoteke. To temelji na dejstvu, da se informacij, skritih znotraj časovnih žigov, ne da razlikovati od časovnih žigov, ustvarjenih z običajno uporabo operacijskega sistema. To zahtevo imenujemo nerazpoznavnost.

Kriptografija nam zagotavlja dodatno zaščito v primeru, da neka tretja oseba odkrije steganografsko plast podatkov. Zato s pomočjo učinkovite simetrične tokovne šifre poskrbimo za šifriranje podatkov, ki jih nameravamo skriti znotraj časovnih žigov. Simetrični ključ je izpeljan iz gesla, ki ga mora uporabnik zagotoviti za vsako šifriranje ozziroma dešifriranje.

Ob uporabi takšne tehnike skrivanja podatkov v številnih scenarijih predvidevamo, da je datotečni sistem v celoti dostopen, tj. uporabnik lahko kadar koli dostopa do datotek in imenikov ter ustvarja, preimenuje ali briše datoteke in mape.

4.1 Sistemskie zahteve

Kot nosilec za prikrivanje podatkov bomo uporabili metapodatke datotečnega sistema, natančneje nanosekundni del časovnega žiga, pri čemer ne bomo uporabljali kakršnih koli drugih podatkovnih struktur, npr. dodatne datoteke z imenom vseh inodov, ki hranijo skrite podatke (uporaba takšne datoteke je sumljiva, tudi če je zašifrirana).

Privzemimo, da je sistem sposoben samodejno določiti primerne časovne žige in sposoben samodejno preveriti, ali je v trenutnem datotečnem sistemu na voljo zadostna količina časovnih žigov. Za skrivanje bomo uporabljali samo že alocirane inode, saj so podatki o nealociranih inodih sestavljeni iz samih ničel. Podatki skriti znotraj nealociranih inodov, bi tekompri preiskave sprožili sum o skrivanju podatkov.

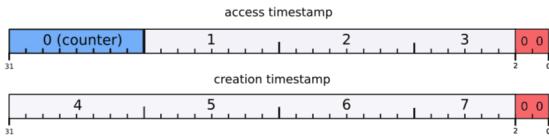
Za dodatno varnost občutljivih podatkov se poleg steganografske tehnike uporablja tudi šifriranje podatkov. Skrivno sporočilo, skrito v predlaganem steganografskem kanalu, bo zato zelo težko razkriti tekom forenzične preiskave.

Informacije bodo shranjene v večih steganografskih nosilcih, zato bo sporočilo razdrobljeno po celotnem datotečnem sistemu, kar bo forenzično preiskavo še bolj otežilo. Dostop do skritih podatkov je mogoč le v primeru, da je uporabniku znan kriptografski ključ za šifriranje in natančna lokacija skritih podatkov znotraj datotečnega sistema. Do teh podatkov lahko uporabnik dostopa zgolj s poznanim gesлом, ki ga navede pred pričetkom postopka.

4.2 Podatkovna enota za skrivanje podatkov

Slika 3 prikazuje 32-bitna polja *i_atime_extra* in *i_crttime_extra* ext4 časovnih žigov. Za skrivanje podatkov bomo uporabljali takšno podatkovno strukturo velikosti od 2 do 30 bitov, brez uporabe spodnjih dveh bitov. Vsak zapis v inod tabeli je sestavljen iz časovnega žiga, ki predstavlja čas ustvarjanja datoteke in časovnega žiga, ki predstavlja zadnji čas dostopanja do datoteke. Ta dva podatka bomo uporabili za nadaljnje skrivanje podatkov. Ker želimo skriti podatke, ki so lahko veliko večji od prostora, ki ga zaseda nanosekundni časovni žig (60 bitov = 7,5 bajtov), bomo vhodno sporočilo razdelili na več delov in vsak del skrili znotraj novega inoda.

Med procesom skrivanja podatkov se širje bajti podatkov zapišejo na mesto *i_atime_extra* in *i_crttime_extra* ext4 časovnih žigov, pri čemer se upošteva ustrezni odmik. Zato je



Slika 3: Prikaz skrivanja podatkov v nanosekundnem delu ext4 časovnih žigov.

potrebno zagotoviti, da se v 4. in 8. bajtu bloka za shranjevanje uporabi največ 6 bitov (namesto 8 bitov). To pomeni, da moramo spodnja dva bita nastaviti na nič. Časovni žigi se namreč berejo kot little-endian, zato bi uporaba vseh 8 bitov v 4. in 8. bajtu lahko sprožila napako povezano z datumi po letu 2038, opisano v poglavju 3.

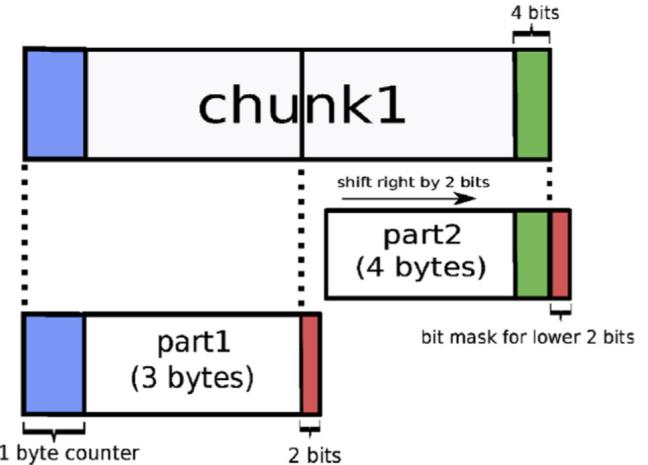
4.3 Obdelava podatkov

Ob skrivanju in obnavljanju skritih podatkov moramo zagotoviti pravilno zaporedje zapisovanja in branja posameznih kosov v in iz datotečnega sistema. Zato potrebujemo urejen seznam inodbv, ki jih lahko uporabimo kot podatkovne nosilce.

Neuner je v [7] predlagal, da se logični vrstni red zagotovi tako, da se med postopkom skrivanja in obnavljanja podatkov iz sistema pridobi seznam vseh datotek, ki so razvrščene po času njihovega nastanka. Takšen pristop se ne zdi učinkovit v sistemih z veliko količino podatkov, saj je najprej potrebno pridobiti metapodatke posameznih datotek in jih nato na podlagi teh informacij pravilno razvrstiti. Namesto tega je smiselno uporabiti podatkovne strukture datotečnega sistema, ki že same po sebi skorajda samodejno zagotavljajo pravilni vrstni red. V primeru NTFS datotečnih sistemov je zato smiselno uporabiti sekvenčno obdelavo zapisov MFT. Podobno kot zapisi MFT se lahko v ext4 datotečnih sistemih pri skrivanju in obnavljanju podatkov uporabi zaporedne številke inodov v inod tabelah.

Za zaznavanje naslednjega veljavnega časovnega žiga med postopkom obnovitve podatkov mora med shranjenimi podatki obstajati neka referenčna točka (npr. nek bitni vzorec ali števec). Zato v vsakem kosu podatkov prvi bajt shranjenih podatkov hrani podatek, ki služi kot števec, kot je prikazano na sliki 3. To pomeni, da so vsi časovni žigi, ki se uporabljajo za skrivanje podatkov ustrezno oštevilčeni. Ker se nealocirane inod med postopkom shranjevanja preskoči, lahko števec uporabimo tako, da ta določi, kateri od naslednjih inodov vsebuje veljavne skrite podatke. Zaradi uporabe indeksnega bajta podatkov nam v posameznem inodu za shranjevanje uporabniške podatke preostane 6,5 bajtov. Najmanjša možna vrednost števca 0x00 poleg začetnega indeksa dodatno vsebuje še informacijo o številu uporabljenih inodov, dolžino celotnega sporočila in kodo za odpravljanje napak. Te informacije so bistvene za pridobivanje skritih podatkov. Uporaba samo enega bajta za shranjevanje indeksa nam dovoljuje, da lahko podatek naenkrat razpršimo med 255 inodov. V primeru, da se pojavi overflow, števec ponastavimo na začetno vrednost 0x00, kar pomeni, da si lahko različni števci v več inodih delijo isto vrednost. V primeru prepisa začetnega inoda se lahko za pridobitev ustreznih informacij, kot je npr. dolžina sporočila, alternativno uporabi tudi 256. zaporedni inod.

Ker računalniki podatke shranjujejo kot zaporedje bitov morajo biti podatki predobdelani tako, da zadnja, spodnja, dva bita vsebujejo ničelno vrednost. Zato se ob branju vhodnih podatkov te najprej razdeli na več 7-bajtnih kosov, kjer se dva kosa razbitih podatkov prekrivata v 1 bajtu podatkov. To pomeni, da se zadnji bajt prvega kosa ujema s prvim bajtom drugega kosa 7 bajtov podatkov in tako naprej. Ta korak je pomemben za nadaljnje operacije, saj lahko v vsakem kosu podatkov s pomočjo operacij premika in logičnih operatorjev zavrzemo 4 bite podatkov ne da bi pri tem izgubili del podatkov.

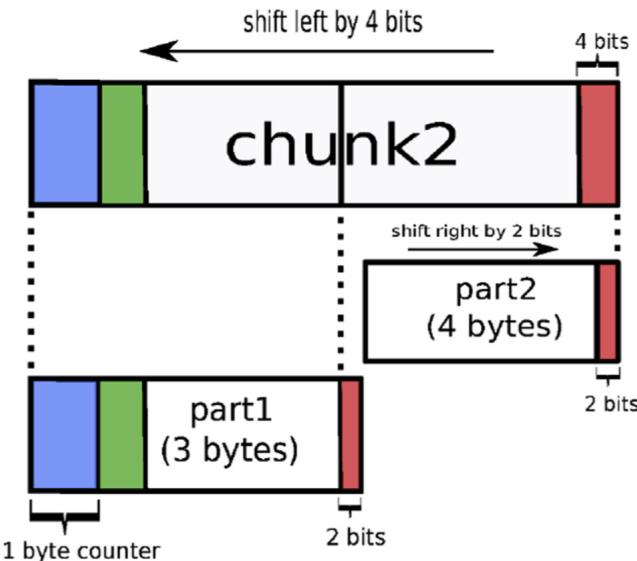


Slika 4: Prvi korak predprocesiranja podatkov.

Na sliki 4 in sliki 5 je prikazana obdelava podatkov, ki se zgoditi vsakič, ko se prebere 14 bajtov podatkov oz. dvakrat po 7 bajtov podatkov. Po predobdelavi se podatki ujemajo s strukturo prikazano na sliki 3. Zato moramo za vsak veljaven datum vstavite štiri ničelne bite - v vsakega od 7 bajtnih kosov po 2 bita na mesti spodnjih dveh bitov. Celotna dobljena podatkovna struktura je sestavljena iz enega kosa podatkov, ki vključuje 2 ničelna bita, indeksnega bajta podatkov in preostalih podatkov, ki jih želimo skriti. Skupna velikost takšne podatkovne strukture znaša 8 bajtov.

4.4 Šifriranje podatkov

Pred vstavljanjem informacij v časovne žige se izvede dodatno šifriranje podatkov. Ta pristop zagotavlja dodatno zaščito v primeru, da je razkrit steganografski kanal v časovnih žigih. Z uporabo števca se podatki razpršijo po celotni tabeli inodov, kar poveča možnost odkritja skritih informacij. Šifrirni proces nam zagotavlja generiranje naključnih časovnih žigov, s čimer preprečimo zaznavanje morebitnih vzorcev pri razporejanju podatkov, niti ne razkrijemo skritih informacij osebam, ki ne pozna kriptografskega ključa. Za šifriranje je bolj primerna uporaba tokovne šifre, saj se lahko za razliko od blokovne šifre, uporablja šifra brez fiksne velikosti za vhodni blok podatkov. To pomeni, da je tokovna šifra sposobna obdelati poljubno dolgo zaporedje vhodnih znakov. Uporaba tokovne šifre nam zagotavlja, da je dolžina vhodnega in izhodnega sporočila ekvivalentna, saj je vsak vhodni znak vezan na pripadajoč šifrirni znak. Tokovna šifra je zato manj občutljiva v primeru manjkajočih podatkov kot npr. blokovna šifra, kjer se lahko zgodi, da v primeru manjkajočih podatkov ne moremo obnoviti celotnega bloka



Slika 5: Drugi korak predprocesiranja podatkov.

podatkov.

Med procesom skrivanja in obnovitvljana informacij se s pomočjo razpršitvene SHA-256 funkcije izračunava enolična zgoščena vrednost, ki je odvisna od vnesenega gesla. Izračunana enolična zgoščena vrednost hrani podatke o številki prvega inoda v katerem so skriti podatki. To programu omogoča pričetek skrivanja podatkov v naključno izbranem inodu, glede na vneseno geslo so torej podatki razpršeni med različne inode. Če v datotečnem sistemu ne obstaja dovolj alociranih inodov, se program zažene v inodu številka 12, saj je to običajno prvi uporabni alocirani inod.

4.5 Kod za odpravljanje napak

V splošnem se lahko ob uporabi podatkovnih struktur datotečnega sistema za namene skrivanja podatkov zgodi, da datotečni sistem pokvari skrito vsebino. Ob uporabi datotečnega sistema, ki je v celoti dostopen uporabniku, se lahko zgodi, da se npr. spremeni vsebina neka datoteke ali pa njeni metapodatki, kar se odraža tudi v delni spremembi skritih podatkov. Z brisanjem starih datotek in ustvarjanjem novim se lahko zgodi, da se obenem izbrišejo tudi skrite informacije. S pomočjo uporabe koda za odpravljanje napak poskrbimo za preverjanje celovitosti podatkov in obnovo spremenjenih ali izbrisanih delov podatkov. V praktični implementaciji se za te namene lahko npr. uporablja odprtakodna Python knjižnica algoritma Reed-Solomon.

5. IMPLEMENTACIJA

5.1 Pomembni podatki za proces skrivanja podatkov

Za skrivanje podatkov potrebujemo naslova nanosekundnih polj časovnega žiga za hranjene časa ustvarjanja datoteke in časovnega žiga za hranjenje časa zadnje ga dostopa datoteke.

Najprej se iz nadbloka preberejo podatki:

- število vseh blokov st_blk ,

- število blokov v eni bločni skupini $st_blk_v_skp$ in
- velikost deskriptorja skupin.

Nato s pomočjo formule izračunamo število bločnih skupin

$$st_blk_skup = \frac{st_blk}{st_blk_v_skp} \quad (1)$$

in iteriramo skozi tabelo deskriptorjev bločnih tabel ter za vsako bločno skupino shranimo naslovni zamik tabele in bitne preslikave inodov.

Nato za vsak inod izračunamo naslednje:

1. bločno skupino inoda:

$$blk_skp_addr = \frac{st_inod - 1}{st_inod_v_skup} \quad (2)$$

2. index inoda znotraj bločne skupine:

$$inod_idx = (st_inod - 1) \bmod st_inod_v_skup \quad (3)$$

3. z uporabo shranjenega seznama zamikov inodov izračunamo zamik $offset_inodtable$

4. zamik inoda

$$offset_inod = inod_idx \cdot inod_size \quad (4)$$

Nato pa lahko izračunamo zamika časovnih žigov:

$$offset_w_size = offset_inodtable \cdot blk_size \quad (5)$$

$$offset_access = offset_w_size + offset_inod + 140 \quad (6)$$

$$offset_creation = offset_w_size + offset_inod + 148 \quad (7)$$

Program nato preveri alociranost inodov. Če je alociranih premalo inodov, o tem obvesti uporabnika in se ustavi.

5.2 Skrivanje informacij

Uporabnik programu poda pot do datoteke D , ki jo želi skriti in ključ za šifriranje podatkov. Program izvede naslednje:

1. Izračuna kod za odpravljanje napak $E = ECC(D)$;
2. pridobi napravo, na kateri se datoteka D nahaja;
3. iz ext4 nadbloka prebere potrebine podatke, opisane v 5.1;
4. z uporabo zgoščene vrednosti gesla izračuna prvi inod in preveri alociranost inodov. Število n potrebovanih inodov je odvisno od dolžine E ;

5. E se razdeli na n 7 bajtov velikih delov ($B_0 \dots B_n$), pri tem se pri vsakih 2 sledečih delih 1 bajt prekriva;
6. Pred vsak del B_i se doda števec velikosti 1 bajta, ki predstavlja indeks bloka, tako da je vsak del zapisan kot ($\text{števec}, B_i$). Ob dosegu največje vrednosti $0xFF$, se števec nastavi na $0x01$. Prvi in vsak 256. del pa je sestavljen kot $(0x00, n, \text{len}(E))$, torej vse skupaj izgleda kot: $(0x00, n, \text{len}(E)), (0x01, B_1), (0x02, B_2) \dots (0xFF, B_{255}), (0x00, n, \text{len}(E)), (0x01, B_{256}) \dots (n, B_n)$;
7. vsak del se skupaj s števcem šifrira s simetrično tokovno šifro z uporabo gesla vnesenega v program. Rezultat so šifrirani deli velikosti 8 bajtov ($C_1 \dots C_n$);
8. vsakemu šifriranemu delu se izničita najnižja dva bita;
9. Po potrebi se zadnji šifrirani del zapolni z ničelnimi bajti;
10. prvi 4 bajti se zapišejo v nanosekundni del časovnega žiga za čas dostopa datoteke, drugi 4 bajti pa v nanosekundni del časovnega žiga za čas stvaritve datoteke.

Uporabljajo se samo alocirani inodi, ker so nealocirani nastavljeni na 0.

5.3 Pridobivanje skritih podatkov

Uporabnik programu poda samo geslo. Nato program:

1. pridobi napravo, na kateri se datoteka D nahaja;
2. iz ext4 nadbloka prebere potrebne podatke, opisane v 5.1;
3. z uporabo zgoščene vrednosti gesla se izračuna prvi inod, ki hrani skrite informacije;
4. če se prvi inod začne z $0x00$ potem se pridobi število uporabljenih inodov n in velikost skritega sporočila E ;
5. prebere se podatke iz izračunanih zamikov za časovna žiga. Bere se samo podatke z naraščajočim števcem;
6. v spodnja dva bita vsakega dela se prepišejo originalni podatki;
7. zadnjemu delu se odstrani zapolnenje;
8. vsi kriptirani deli so dekriptirani z simetrično tokovno šifro in podanim gesлом;
9. vsi vmesni deli z $0x00$, bajti s števcem in vsi podvojeni bajti so odstranjeni;
10. datoteko povrne s kodom za odpravljanje napak: $D = ECC(E)$ in jo zapiše na disk.

5.4 Popravek preizkusne vsote tabel inodov

Ker se spremenijo podatki inoda se preizkusne vsote posodobljenih inodov ne ujemajo s preizkusnimi vsotami shranjenimi v tabelah. Ker podatke hočemo skriti, je preizkusnim vsotam potreben popraviti vrednosti.

6. OCENA PREDLAGANEGA NAČINA SKRIVANJA PODATKOV

Göbel in Baier sta predlagan način skrivanja podatkov ocenila z vidika nerazpoznavnosti skritih podatkov, pravilnosti, robustnosti in možne kapacitete [5]. Ker program ni na voljo na platformi GitHub, navedenih rezultatov nismo mogli preveriti na naših računalnikih.

6.1 Nerazpoznavnost

Ocenjevanje nerazpoznavnosti skritih podatkov nam pove kako so skriti podatki podobni pravim vrednostim časovnih žigov. Zaradi simetrične tokovne šifre sta *Göbel in Baier* predpostavila enakomerne porazdelitev časovnih žigov, ki vsebujejo skrite podatke [5]. Za kvantifikacijo nerazpoznavnosti izračunajo informacijsko entropijo tako za nemodificirane časovne žige kot tudi za tiste kjer so nekateri modificirani, potem pa jih primerjajo.

Pri ocenjevanju sta se omejila na preverjanje odstopanja spodnjih 10 bitov časovnega žiga, torej najbolj natančnega dela, kjer se dva časovno zaporedna žiga lahko razlikujeta od 0 do 1023 ns . Če je pridobljena porazdelitev enakomerna na 10 bitih potem predpostavlja enakomerne porazdelitev pri vseh 30 bitih.

Za pridobitev zadostnega števila podatkov je bil test izveden večkrat, skrivali pa so 357574 bajtno datoteko. Najprej so zgenerirali $100000, 250000$ in 500000 datotek v naključnem časovnem razmaku med $1\text{ ns} - 1\text{ }\mu\text{s}$. Nato so za vsakega izmed teh treh primerov:

1. prebrali vrednosti spodnjih 10 bitov nanosekundnega dela časovnih žigov in izračunali njihovo entropijo,
2. skrili datoteko v časovne žige
3. ponovno prebrali vrednosti spodnjih 10 bitov nanosekundnega dela časovnih žigov in izračunali njihovo entropijo.

Pri pridobljenih entropijah so izračunali aritmetično povprečje in standardni odklon tako za primere brez skritih podatkov kot za primere s skritimi podatki.

	st. datotek	100000	250000	500000
Kontrola:	Odklon	0.0389	0.0547	0.0028
	Povprecje	9.9434	9.9672	9.9928
Skrivanje datoteke:	Odklon	0.0272	0.0053	0.0015
	Povprecje	9.9069	9.9795	9.9915

Tabela 2: Rezultati testa entropije pred skrivanjem datoteke in po skrivanju datoteke. Podatki iz [5]

Iz rezultatov, prikazanih na tabeli 2 so prepoznali, da porazdelitev vrednosti pravih časovnih žigov ne odstopa od enakomerne porazdelitve in da so posledično s skrivanjem datoteke spremenjeni časovni žigi dovolj nerazpoznavni od avtentičnih.

6.2 Pravilnost

Z ocenjevanje pravilnosti delovanja predlaganega načina za skrivanje datotek v časovnih žigih *Göbel in Baier* v [5] predstavita več testov.

Integriteto lahko testiramo z uporabo zgoščevalne funkcije SHA-256. Najprej se izračuna zgoščena vrednost datoteke pred skrivanjem, nato pa se izračuna zgoščena vrednost pridobljene datoteke iz časovnih žigov. Če sta zgoščeni vrednosti enaki, je pridobljena datoteka zagotovo enaka izvorni.

Da program ni prepisal spodnjih dveh bitov nanosekundnih časovnih žigov lahko preverimo z uporabo *Linux* ukazov *stat* in *debugfs*, kjer se prepričamo, da leto v datumu ni previšoko.

Prav tako je treba preveriti, da sta se sporočilo in števec zapisala zašifrirana.

Z uporabo *e2fsck* pa preverimo ali je datotečni sistem v konsistentnem stanju [1], torej tako preverimo, ali je program pravilno preračunal in zamenjal preizkusne vsote uporabljenih inodov.

6.3 Robustnost

Če je volumen priklopljen z možnostjo *noatime* (kar pomeni *nodiratime*), lahko vgnezdene informacije prepišemo le ročno, ker se časovni žigi ne posodabljajo več. Ker smo prek ukaza cat dostopali do več datotek nosilcev, so bili posodobljeni časovni žigi dostopa. Poleg tega je bilo z ukazom rm ročno odstranjenih več datotek. Nove datoteke so bile ustvarjene z ukazom touch. Ukaz *stat* kaže, da so bili časovni žigi dostopa in kreiranja spremenjeni.

Največje število prepisanih časovnih žigov je odvisno od števila dodatnih redundantnih bajtov. Če se pričakuje velika količina prepisanih informacij (npr. V večuporabniškem sistemu), lahko raven redundancy povečamo. Če pa ne pride do prepisovanja časovnih žigov, lahko metodo za odpravljanje napak v celoti preskočimo, kar povzroči povečanje količine podatkov, ki jih je mogoče skriti. V izvedbi PoC so redundantne informacije namenoma porazdeljene po celotnem datotečnem sistemu. Odvisno od ECC in uporabe datotečnega sistema, bi lahko bile redundantne informacije shranjena tudi na fiksni lokaciji (npr. inodi nespremenljivi/neizbrisljivi datotek), da je ne bi bilo mogoče izgubiti.

6.4 Razpoložljiv prostor

Količina uporabnega prostora je odvisna od skupne velikosti datotečnega sistema in števila inodov. Za analizo števila razpoložljivih inodov smo ustvarili testne ext4 volumne z velikostmi 1 GB, 10 GB, 50 GB in 100 GB. Tabela 3 prikazuje število razpoložljivih inodov glede na velikost volumna.

Tabela 4 kaže, da predlagana tehnika skrivanja podatkov hitro doseže limite. Za prej skrito bitmap datoteko je bilo potrebnih 55.228 inodov, ki že porabijo 90 odstotkov največje razpoložljivih inodov na volumnu z velikostjo 1 GB. Za skrivanje datotek v velikosti 1 MB je že potrebnih 183.190 inodov, to pomeni, da bi bila potreben 3 GB velik volumen, poln podatkov. Pri večjih datotekah količina zahtevanih inodov hitro preseže milijon. Zato ta antiforenzična tehnika ni namenjena skrivanju velikih datotek, kot so slikovne ali vi-

deo datoteke. Namesto tega lahko skrijemo majhne besedilne datoteke.

Velikost	Skupno število inodov
1 GB	61057
10 GB	610.801
50 GB	3.055.617
100 GB	6.111.233

Tabela 3: Število možnih inodov glede na velikost volumna.

Velikost vhoda	Potrebno število inodov
1 KB	186
100 KB	18.325
500 KB	91.595
1 MB	183.190
5 MB	915.923
10 MB	1.831.846
50 MB	9.159.219

Tabela 4: Potrebno število inodov glede na velikost vhoda.

7. VPLIVANJE NA POTEK FORENZIČNE PREISKAVE

Zaradi uporabe šifriranja vhodnih podatkov se za obnovo podatkov v fazi forenzične preiskave ne moremo zanašati na entropijo odkritih podatkov. V primeru izpustitve koraka šifriranja pa bi lahko s statistično analizo prepoznali morebitne vzorce v podatkih. Če obstajajo varnostne kopije datotečnega sistema se lahko za označevanje skritih informacij uporabi primerjavo kopije datotečnega sistema s trenutno različico z namenom odkritja morebitnih razlik. V primeru uporabe datotek operacijskega sistema, kot podatkovnih nosilcev, se lahko v fazi forenzične preiskave primerja kronološko zaporedje časovnih žigov namestitvenih datotek s spremenjenimi časovnimi žigi in tako odkrije morebitne nedoslednosti.

Ob razvoju rešitve moramo poskrbeti, da v sistemu ne pustimo morebitnih sledi o uporabi opisanega orodja. To pomeni, da ne želimo uporabljati namestitvenih datotek, v sistemu pustiti morebitnih sledi izvajanja orodja ali kakršnih koli drugih artefaktov, ki bi dokazovali, da je bilo uporabljen anti-forenzično orodje. Po zaključku skrivanja podatkov je koristno preveriti, ali so vsi časovni žigi veljavni in smiseln kronološko urejeni. Takšnih neskladnosti se lahko znebimo s prilagoditvami vrednosti časovnih žigov, ki imajo natančnost višjo od nanosekund.

8. ZAKLJUČEK

Analizirali smo možnost uporabe časovnih žigov datotečnega sistema ext4 kot steganografskega kanala za prikrivanje podatkov. Podrobneje smo opisali proces skrivanja in obnavljanja informacij v ext4 datotečnem sistemu. Izkazalo se je, da je informacije možno skriti znotraj časovnih žigov datotečnega sistema ext4 katerih granulacija je manjša od sekundnih podatkov. Zaradi uporabe dvostopenjskega postopka skrivanja podatkov, ki temelji na steganografiji in kriptografiji, je skrito vsebino statistično nemogoče razlikovati od običajnih, sistemskih, vrednosti časovnih žigov.

Po objavi izvorne kode projekta na platformi GitHub bi bilo smiselno analizirati teoretično delovanje skrivanja podatkov in to tudi praktično preizkusiti.

9. Literatura

- [1] e2fsck(8): check ext2/ext3/ext4 file system - linux man page.
- [2] R. Anderson, R. Needham, and A. Shamir. The steganographic file system. pages 73–82, 1998.
- [3] K. Conlan, I. Baggili, and F. Breitinger. Anti-forensics: Furthering digital forensic science through a new extended, granular taxonomy. *Digital Investigation*, 18:S66 – S75, 2016.
- [4] K. D. Fairbanks. An analysis of ext4 for digital forensics. *Digital Investigation*, 9:S118 – S130, 2012. The Proceedings of the Twelfth Annual DFRWS Conference.
- [5] T. Göbel and H. Baier. Anti-forensics in ext4: On secrecy and usability of timestamp-based data hiding. *Digital Investigation*, 24:S111 – S120, 2018.
- [6] A. Mathur, M. Cao, S. Bhattacharya, A. Dilger, A. Tomas, and L. Vivier. The new ext4 filesystem: current status and future plans. 2:21–33, 2007.
- [7] S. Neuner, A. G. Voyatzis, M. Schmiedecker, S. Brunthaler, S. Katzenbeisser, and E. R. Weippl. Time is on my side: Steganography in filesystem metadata. *Digital Investigation*, 18:S76–S86, 2016.
- [8] M. Rogers and M. Lockheed. Anti-forensics. *Lockheed martin. San Diego, California. Retrieved from http://cyberforensics. purdue. edu/documents/AntiForensics\LockheedMartin09152005. pdf*, 2005.
- [9] D. J. Wong. Ext4 disk layout - ext4.