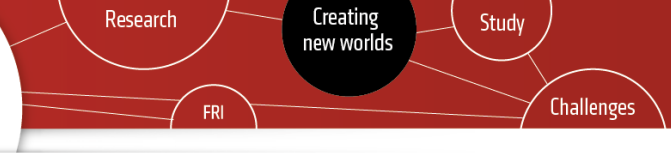
University of Ljubljana
Faculty of Computer and Information Science



Zbornik

Digitalna forenzika

Seminarske naloge, 2015/2016

Ljubljana, 2016

Zbornik
Digitalna forenzika, Seminarske naloge 2015/2016

Editors: Andrej Brodnik, Jon Muhovič, Primož Črnigoj, Jure Kolenko, študenti

Ljubljana : Univerza v Ljubljani, Fakulteta za računalništvo in informatiko, 2016.

Kazalo

Uvod

Digitalna forenzika je veja forenzične znanosti. Ukvarja se z zbiranjem in preiskavo digitalnih dokazov, ki jih lahko najdemo na različnih napravah z digitalnimi podatkovnimi nosilci. Pri izvajanju kaznivih dejanj storilci puščajo veliko sledi na digitalni opremi v njihovi neposredni bližini ter tudi na digitalni opremi na drugem koncu sveta. Okoli nas je veliko kamer, mikrofonov ter tudi brezžične komunikacije. Tako lahko že ob sprehodu po stavbi naš mobilni telefon zazna več brezžičnih usmerjevalnikov omrežnega prometa ter zabeleži našo prisotnost. Storilec lahko izbriše podatke, a so ti še vedno prisotni na nosilcu. Če se kaznivo dejanje izvaja s pomočjo računalnika, lahko dokazi o tem ostanejo na oddaljenih strežnikih. S prebojem tako imenovanega interneta stvari naprave hranijo podatke o tem, kdaj je bila nazadnje skuhana kava, kdaj je bil nazadnje vožen avtomobil, kolikokrat in kdaj so se odprla vrata pametne hiše. Takšnih naprav je iz dneva v dan več in iz tega izhaja tudi naraščajoča potreba po znanju s tega področja. Hitra rast in razvoj digitalnih naprav zahteva pogosto osveževanje in posodobitev znanja o ravnanju z napravami in tehnikah pridobivanja dokazov.

V zborniku so zbrane seminarske naloge magistrskih študentov Fakultete za računalništvo in informatiko Univerze v Ljubljani 2015/2016, ki smo jih izdelali pri predmetu Digitalna forenzika. V seminarskih nalogah smo pregledali nekaj najnovejših člankov (2015/2016) z različnih področij digitalne forenzike ter napisali obnove ter mnenja o njih. Nekatere rešitve,predlagane v člankih, so bile tudi preizkušene. Po oddaji lastnih seminarskih nalog je vsak pregledal in ocenil še dve seminarski nalogi svojih kolegov in tako tudi sam dobil recenzijo svoje seminarske naloge. Nato smo seminarske naloge ustrezno popravili in pripravili 15 minutne predstavitve, ki smo jih izvedli na zadnjih dveh predavanjih predmeta.

Zbornik je razdeljen na pet sklopov: Forenzika notranjega pomnilnika, Omrežna forenzika in forenzika mobilnih naprav, Forenzika slik, Forenzika podatkovij in triaža ter Forenzika e-pošte in spletnih storitev. Sklop Forenzika notranjega pomnilnika vsebuje naloge, ki predstavijo načine zajemanja podatkov iz pomnilnika digitalnih naprav z različno strojno(npr. grafična procesna enota) in pogramsko opremo na različnih platformah. V sklopu Omrežna forenzika in forenzika mobilnih naprav najdemo nalogi, ki opišeta uporabo različnih senzorjev na pametnih telefonih kot forenzični dokaz in prikažeta različne varnostne pomanjklivosti popularnih aplikacij za takojšnje sporočanje (Viber, WhatsApp, Instagram ipd.). V sklopu Forenzika slik se nahajajo naloge, ki se ukvarjajo s skrivanjem sporočil znotraj slik, detekcijo naknadnih sprememb slik in določanjem modela kamere, s katero je bila posneta slika. Sklop Forenzika podatkovij in triaža se ukvarja z višjenivojskimi pristopi k pregledovanju dokaznega gradiva, metodami za hitro pregledovanje večjih količin pomnilnika in postopki za forenzično analizo podatkovnih baz. Zadnji sklop, Forenzika e-pošte in spletnih storitev, pa vsebuje naloge, ki pregledajo področje zasebnosti pri forenzični analizi elektronske pošte, določanje avtorstva elektronskega sporočila, analize digitalnih artefaktov ter analize vedenjskih vzorcev pri spletnih zalezovalcih.

Primož Črnigoj in Jon Muhovič

Poglavje 1

Povzetki

1.1 Raziskava vpliva zlonamerne kode, ki uporablja grafično procesno enoto, na forenzično analizo

V tej raziskavi raziščemo možnosti uporabe grafičnih kartic za skrivanje zlonamerne aktivnosti. Analiziramo delovanje grafične kartice kot periferne enote v računalniškem sistemu in izpostavimo razlike v pogledu na naslovni prostor s stališča centralne procesne enote in grafične procesne enote. Predstavimo štiri možne scenarije zlorabe pomanjkljivosti grafičnih kartic in prikažemo njihovo praktično implementacijo na Intelovi platformi HD Graphics.

1.2 Volatile memory forensics and rootkit detection on OS X

The first part of our study reviews the article of A. Case titled Advancing Mac OS X rootkit detection, alongside three important related studies in this field. It highlights the increasing importance of both OS X and volatile memory analysis in forensic science as valuable potential evidence. It focuses on rootkits as an increasingly popular yet under-researched form of malware and reviews some rootkit detection techniques. The Volatility framework and its rootkit detection plugins for Mac OS X are discussed in detail. In the second part our study describes an experiment in which we analyzed a RAM dump originating from a sample OS X device. A variety of tools were used to extract sensitive data from the system key chain and an application specific data store. Such data is shown to potentially be of vital importance to a forensic specialist.

1.3 Pridobivanje vsebine pomnilnika na napravah z operacijskim sistemom Android z uporabo protokolov za posodabljanje strojne programske opreme

V digitalni forenziki že nekaj časa vse bolj pomembna postaja forenzika mobilnih naprav. Pametni telefoni ne hranijo več le podatkov o klicih, temveč tudi uporabnikovo pošto, slike, lokacijo, koledar, zgodovino brskanja, osebne in finančne podatke ter uporabniška gesla do drugih storitev. Skratka, ogromno podatkov o uporabnikovem početju, ki bi lahko služila kot dokazno gradivo na sodišču.

Skupaj z vse pomembnejšo vlogo mobilnih naprav pa se je izboljševala tudi njihova varnost. Pridobitev podatkov z mobilnih naprav je čedalje težja in potrebno je iskanje novih metod, ki lahko zaobidejo varnostne ukrepe.

Seung Jei Yang in soavtorji z inštituta ETRI (angl. Electronics and Telecommunications Research Institute) so v članku z naslovom New acquisition method based on firmware update protocols for Android smartphones predstavili novo metodo za pridobitev vsebine pomnilnika mobilne naprave z operacijskim sistemom Android brez izgube oziroma spremembe celovitosti podatkov. Z obratnim inženirstvom (angl. reverse engineering) postopka posodabljanja strojne programske opreme so odkrili novo metodo zajemanja, s katero lahko s pomočjo protokolov, ki jih sicer uporabljajo proizvajalci mobilnih naprav, neposredno dostopajo do pomnilnika.

Pristop se izkaže za uspešnega in se izogne večini težav, ki se pojavljajo pri zajemanju podatkov z obstoječimi metodami.

1.4 Reliable and Trustworthy Memory Acquisition on Smartphones

More and more malicious code is emerging every day and an insignificant chunk of it is targeting smartphones. The malware and its effects on a system must first be analysed in order to combat it effectively. In this paper we review TrustDump, a mechanism for reliably obtaining the contents of the RAM and CPU registers of a possibly compromised operating system. It is based on ARM's TrustZone technology, where the OS being analysed is running in the normal world and the analysis tools are running in the secure world. The latter has higher privileges and is separated from the former on the hardware level. In this work we describe the TrustDump mechanism and overview some possible alternatives.

1.5 Senzorji na pametnih telefonih kot digitalni dokaz

Senzorji v pametnih telefoni omogočajo nove priložnosti za iskanje dokazov v digitalni forenziki. V pametne telefone se vgrajuje vedno več različnih senzorjev, ki beležijo različne podatke o stanju telefona. Podatki, ki jih beleži pametni telefon so v digitalni forenziki lahko uporabni npr. pri potrjevanju ali zavračanju alibija ali pri dokazovanju dejavnosti oz. aktivnosti osumljenca. Za pregled podatkov na pametnih telefonih digitalni forenziki uporabljajo za ta namen izdelano programsko opremo. V tem članku bomo predstavili idejo o zajemu podatkov različnih senzorjev, kako se te podatke ustezno zajame in pregleda, ter s kakšnimi izzivi se lahko srečamo pri sami pridobitvi podatkov.

1.6 Network and device forensic analysis of Androidsocial-messaging applications

This paper is a summary of the experiments done to research the security of 20 Android applications. Its main focus is set on reconstructing user related data after the use of social-messaging applications. According to their results, even entire message contents were successfully reconstructed from 16 of the 20 applications tested, which shows that security and privacy measures employed by these applications are poor. Besides

security it also focuses on forensic analysis of Android smartphones. The paper also describes the methods and results achieved by related work.

1.7 Določanje modelov digitalnih kamer s podporo neznanih modelov

Določanje modela digitalne kamere na podlagi značilk izluščenih iz digitalne fotografije v digitalni forenziki igra pomembno vlogo. Žal pa imajo vse obstoječe tehnike, ki temeljijo na večrazredni klasifikaciji nad izluščenimi značilkami, skupen problem določanja neznanih modelov. Nekatere fotografije so namreč zajete z digitalno kamero, ki sistemu ni bila vnaprej znana, zato obstoječe tahnike takšne modele napačno klasificirajo v najbližji ciljni razred. Avtorji članka v ta namen predlagajo novo tehniko imenovano SCIU (Source Camera Identification with Unknown models), ki je sposobna tako zaznati ali gre za neznan model kamere, kot tudi razlikovati med znanimi modeli kamer. Predlagana tehnika je sestavljena iz treh delov: 1) detekcija neznanih modelov, 2) razžiritev neznanih modelov, ter 3) klasifikacija v enega izmed K+1 razredov. Detekcija neznanih modelov uporablja metodo k-najbližjih sosedov (KNN) z namenom prepoznavanja primerov neznanih modelov v neoznačeni množici fotografij. Razširitev neznanih modelov nato množico neznanih modelov še dodatno razširi. V zadnji fazi pa združimo množico neznanih modelov (1 razred) ter množico znanih modelov (K razredov), ter le-to uporabimo za učenje na (K+1)-razrednem klasifikatorju. Avtorji članka razvijejo tudi tehniko optimizacije parametov tehnike SCIU. Razvita tehnika je nato preizkušena na množici slik Dresden, kjer avtorji pokažejo dejansko učinkovitost razvite tehnike v primerjavi s tehnikami večrazrednega klasificiranja s podpornimi vektorji (MSVM), binarnega klasificiranja s podpornimi vektorji (BSVM), kombiniranih klasifikacijskih tehnik (CCF) ter klasificiranja DBC (decision boundary carving). Dodatno skušamo avtorjevo delo ter raziskave ponoviti, ter preveriti ali se dobljene rezultati ujemajo s tistimi o katerih poročajo avtorji članka.

1.8 Detekcija modificiranih slik s tehniko lokalnih binarnih vzorcev

Metode, kot je rezanje šivov, ki je zelo priljubljena za spreminjanje velikosti slik, tako da ohranjamo bistvo slike so dandanes zelo priljubljene in jih najdemo praktično v vsakem programu za obdelavo slik. Te metode pa so lahko tudi zlorabljene, tako, da se z njimi spremeni pomen slik. V delu predstavimo različne načine detekcije sprememb v slikah, ki so posledica rezanja šivov. Obstaja veliko različnih načinov detekcije. V našem delu predstavimo tri načine, ki so se v zadnjem času izkazale kot najuspešnejše.

Najprej predstavimo kaj je rezanje šivov, ter kako lahko algoritem uporabimo za namerno spreminjanje vsebine slik. V nadaljevanju implementiramo pristop z binarnimi lokalnimi vzorci in energijskimi značilkami, ki predstavljajo vhod v učni algoritem, s katerim ugotovimo, ali je bila slika modificirana z metodo rezanja šivov. Metodo testiramo pri različnih testnih pogojih.

Uporabili smo protokol testiranja, kot ga uporabljajo v vseh člankih s tega področja, kar omogoča primerjavo med različnimi metodami. Protokol testiranja smo tudi izboljšali, tako da smo testiranja izvedli na način, ki bolje izražajo realno uporabo metod detekcije, saj smo ugotovili, da je protokol, ki ga uporabljajo raziskovalci precej generičen in ne predstavlja realne ocene metod. Poleg tega smo raziskavo naredili na način, ki omogoča enostavno ponovljivost rezultatov.

1.9 JPEG steganografija

V seminarski nalogi orišemo področje steganografije, kratko opišemo JPEG kompresijo in naredimo pregled state-of-the-art metod, ki skušajo statistično neopazno skriti kar se da dolga sporočila v digitalne slike, stisnjene s postopkom JPEG, ena od njih je opisana v obdelanem članku.

1.10 Overview of the Tiered forensic methodology model for Digital Field Triage by non-digital evidence specialists

This paper provides an overview of the "Tiered forensic methodology model for Digital Field Triage by non-digital evidence specialists" paper by Ben Hitchcock, Nhien-An Le-Khac, and Mark Scanlon presented on the DFRWS EU conference on 31.03.2016, as well as a general overview of the digital forensic triage and investigation. The paper deals with new ways (model) to conduct the forensics investigations with focus on budgetary and time (laboratory backlog) constraints. The main proposed change consists of elementary training of front-line personnel in the field of digital triage conducted on-scene or shortly after the evidence collection phase. The paper also presents an overview of implementation of this model in real-world environment (namely Canadian police forces).

1.11 Hitro forenzično slikanje velikih diskov s presejanjem

V tem članku je predstavljen postopek hitrega kloniranja diskov avtorjev Jonathana Griera in Goldena G. Richarda III. V primerjavi z obstoječimi pristopi k reševanju problemov, ki jih predstavljajo vedno večje kapacitete diskov, ta postopek zagotavlja veliko večjo zanseljivost dokaznega gradiva in ponovljivost preiskave. Rezultat slikanja diska s presejanjem je natančna nizko nivojska kopija izbranih sektorjev, shranjena v formatu AFF v3. Na ta način je odpravljena težava forenzike v živo, kjer izgubimo podatke z diska in shranimo le rezultate analize. Zelo pomemben je izbor sektorjev, ki je skoraj popolnoma avtomatiziran in od preiskovalca zahteva le definicijo profila preiskave, ki s preprostimi pravili usmerja iskanje pomembnih sektorjev pri zajemu. Na testnih primerih je opisani postopek dosegel več kot 3-kratne pohitritve z manj kot 5% izgubo dokaznega gradiva.

1.12 Izrezovanje na podlagi izvlečkov podatkovnih sektorjev

Izrezovanje na podlagi izvlečkov je tehnika za odkrivanje prisotnosti določenih datotek na podatkovnih nosilcih s primerjanjem izvlečkov posameznih podatkovnih sektorjev namesto celotnih datotek. Na ta način lahko identificiramo fragmentirane, nepopolne in celo delno spremenjene datoteke. Dosedanji poizkusi izrezovanja na podlagi izvlečkov so bili izvedeni na relativno majhnih množicah podatkov. Avtorji članka izvedejo izrezovanje z bazo, ki obsega približno miljon ciljnih datotek, in pri tem odkrijejo nepričakovano visoko stopnjo napake zaradi ponavljajočih se podatkovnih struktur v dokumentih Microsoft Office in multimedijskih datotekah. V nadaljevanju predstavijo rešitev v obliki dveh algoritmov HASH−SETS in HASH−RUNS, ki omogočata odkrivanje prisotnosti datotek in njihovo rekonstrukcijo s pomočjo baze izvlečkov podatkovnih sektorjev. Pri demonstraciji tehnike si pomagajo z orodjem bulk_extractor, podatkovno bazo hashdb in implementacijo algoritma v jeziku Python. V seminarski nalogi si najprej pogledamo stanje na tem področju,

osnovno idejo izrezovanja na podlagi izvlečkov podatkovnih sektorjev in nato še predlagano rešitev iz članka na primeru.

1.13 Forenzična analiza podatkovne baze skozi pregled notranjih podatkovnih struktur

Dandanes je večina podatkov, s katerimi imamo opravka, shranjena v digitalni obliki. Forenzična analiza je tako največkrat osredotočena na restavriranje digitalnih vsebin in rekonstrukcijo dejanj uporabnika iz samih posnetkov sistema, na katerem je uporabnik izvajal akcije. Predmet raziskave v tem delu so tako programske tehnike restavriranja podatkov iz relacijskih podatkovnih baz (ang. DMBS). Pri tem je bila uporabljena tehnika klesanja podatkov iz datotek.

1.14 Ohranjanje zasebnosti pri forenzični preiskavi sporočil elektronske pošte

Seminarska naloga opiše težave, s katerimi se srečujejo preiskovalci pri analizi sporočil elektronske pošte. Pri tem pogosto nezakonito posežejo v posameznikovo zasebnost. Predstavljen je nov način pregledovanja e-poštnih sporočil, ki s pomočjo kriptografije ohranja zasebnost posameznika, hkrati pa preiskovalcu omogoča opravljanje forenzične preiskave. V seminarski nalogi je opisan takšen način, njegovo delovanje ter kratek povzetek njegove uporabe.

1.15 E-mail avtorstvo na podlagi asociativne klasifikacije

Komunikacija prek elektronske pošte se dostikrat uporablja kot orožje nepridipravov, ki želijo na vsak način izkoristiti socialni inženiring in škodovati ljudem. Protokol za elektronska sporočila ima ta problem, da se lahko kdorkoli predstavlja za neko osebo in vseprek pošilja neka sporočila. Tudi, če izbriše sledi iz glave elektronske pošte ali celo uporabi več računov, lahko, misleč, da ne pusti nobenih sledi, iz sloga pisanja, uporabljanja določenih fraz, ločil in strukture besedila s pomočjo asociativne klasifikacije zmanjšamo število osumljencev in ugotovimo katera oseba je napisala nek mail. Na podlagi značilk pisanja in asociativnih pravil zgradimo klasifikator, ki je v tem članku tudi opisan in opisana je tudi sama uporaba le-tega na realnih podatkih.

1.16 Forenzična analiza digitalnih artefaktov

Forenzična analiza artefaktov storitev v oblaku je še vedno v začetnih povojih. Trenutni pristopi vse preveč sledijo tradicionalnim metodologijam pridobivanja artefaktov na strani klienta. V tem delu, predstavljamo koncept analize artefaktov, ki so na strani storitev v oblaku, to so podatki, ki hranijo aktualno stanje spletnih/SaaS aplikacij. Delovanje spletnih aplikacij se povsem razlikuje od delovanja tradicionalnih, spletne aplikacije med delovanjem prenesejo le nujno potrebno stanje datotek ter ob končanem delu za seboj ne puščajo sledi. Medtem, ko namizne aplikacije hranijo trenutno stanje datotek na lokalnem datotečnem sistemu.

Z uporabo Google Docs spletnih aplikacij predstavljamo povsem drugačno strukturo artefaktov, kot smo jo navajeni na namiznih aplikacijah. Stanje le teh je navadno v obliki popolnih ali delnih zabeležk uporabnikovih dejanj urejanja dokumenta. Zato je tradicionalni pristop zajema stanja artefaktov v času samo po sebi forenzično pomanjkljivo saj ne upošteva potencialno pomembnih informacij o razvoju dokumenta v daljšem časovnem obdobju. Artefakti na strani strežnika nimajo standardizirane oblike, kar vzbuja vprašanja v zvezi z njihovo dolgoročno ohranitvijo in interpretacijo.

1.17 Forenzična raziskava primerov spletnega zalezovanja s pomočjo analize vedenjskih razvidov

Z razvojem spleta in vseprisotnostjo računalništva je spletno zalezovanje postalo resen zločin. Metode analize vedenjskih razvidov pomagajo pri razumevanju zločinca, žrtve prizorišča zločina in dinamike zločina. Spadajo pod deduktivne metode. Statistični podatki kažejo, da so najpogostejše žrtve mlade ženske, motiv zalezovalca pa je največkrat ljubezenski. Predstavljena je forenzična analiza 20 primerov, ki jih je preiskovala Dubajska policija. Proces analize digitalnih podatkov je bil iterativen ter progresiven. Pri primerjavi s statističnimi podatki na vecjih množicah smo našli razlike. Rezultati so pokazali, da metode analize vedenjskih razvidov omogočajo boljše razumevanje in interpretacijo obnašanja žrtve in zločinca. Veliko držav se sprejema dodatne zakone, ki pokrivajo spletno zalezovanje.

Del I

Forenzika notranjega pomnilnika

Raziskava vpliva zlonamerne kode, ki uporablja grafično procesno enoto, na forenzično analizo

Žiga Lesar
Univerza v Ljubljani
Fakulteta za računalništvo in informatiko
1000 Ljubljana
ziga.lesar@lgm.fri.uni-lj.si

Matevž Lenič
Univerza v Ljubljani
Fakulteta za računalništvo in informatiko
1000 Ljubljana
matevz.lenic@gmail.com

POVZETEK

V tej raziskavi raziščemo možnosti uporabe grafičnih kartic za skrivanje zlonamerne aktivnosti. Analiziramo delovanje grafične kartice kot periferne enote v računalniškem sistemu in izpostavimo razlike v pogledu na naslovni prostor s stališča centralne procesne enote in grafične procesne enote. Predstavimo štiri možne scenarije zlorabe pomanjkljivosti grafičnih kartic in prikažemo njihovo praktično implementacijo na Intelovi platformi HD Graphics.

Ključne besede

zlonamerna koda, grafična kartica, GPGPU

1. UVOD

V zadnjih desetletjih smo bili priča velikemu porastu zmogljivosti procesorjev. Toda predvsem zaradi približevanju fizikalnim omejitvam, se ta hitra rast v zadnjih letih vidno upočasnjuje. Ena izmed smiselnih rešitev problema je izraba konceptov paralelnega procesiranja. Vendar smo s klasično procesorsko arhitekturo precej omejeni glede števila procesorskih jeder. Te omejitve so privedle do snovanja povsem drugačnih tehnologij in naprav, npr. grafične kartice, signalni procesorji ipd. V tem besedilu se bomo fokusirali predvsem na grafične kartice ter njihove prednosti in slabosti, predvsem na področju varnosti in digitalne forenzike.

Dolgo časa so se grafične kartice uporabljale za pohitritev upodabljanja in drugih grafičnih opravil. Učinkovitost teh naprav se je v zadnjih letih izkazala za prebojno tudi na mnogih drugih področjih v računalništvu, kot so medicinska vizualizacija, strojno učenje, fizikalne simulacije itd.

V začetku je bil dostop do grafične kartice omejen na zelo specifične programske vmesnike, npr. DirectX in OpenGL, ki pa so bili osredotočeni le na grafične aplikacije. Tudi sama strojna oprema razvijalcem in končnim uporabnikom ni dovoljevala razvijanja svojih lastnih algoritmov za izkoriščanje prednosti grafičnih procesnih enot, zato so se morali zateči k programerskim trikom, ki so pogosto bili tudi zelo odvisni od platforme, na kateri je aplikacija tekla. Dandanes so grafične kartice precej bolj splošne naprave, ki jih lahko povsem po svojih željah programirajo tudi končni uporabniki. Razvoj grafičnih kartic je bil dolga leta prav zaradi fiksne funkcionalnosti (angl. fixed pipeline) usmerjen v povečevanje zmogljivosti strojne opreme. Vendar pa je to odprlo veliko možnosti za zlorabe, predvsem s pojavom t.i. splošnonamenskega računanja na grafičnih karticah (angl. General-purpose computing on graphics processing units - GPGPU) in integriranih rešitev, kot je npr. Intel HD Graphics.

Raziskave na temo zlonamerne kode, ki lahko izkorišča grafične kartice so precej redke, vendar pa so redki tudi primeri take zlorabe v praksi. Obstajajo sicer primeri napadov, kjer je grafična kartica glavna tarča, toda v tem besedilu se bomo osredotočili le na napade pri katerih grafična kartica nastopa kot orodje. Sodeč po članku [2] je po navedbah avtorjev znan le en tak primer zlorabe, opisan v [1].

V tem članku bomo raziskali različne tehnike skrivanja aktivnosti in izogibanja varnostnim mehanizmom centralnih procesorskih enot in operacijskega sistema.

2. PREGLED PODROČJA

Uporaba grafičnih kartic za izvajanje zlonamerne kode je precej teoretično področje, ki je zaenkrat še po večini omejeno na eksperimente izvedene v kontroliranem raziskovalnem okolju. Do sedaj je bil na odprtih sistemi odkrit le en primer zlorabe, in sicer program BadMiner, ki uporablja grafično kartico za rudarjenje digitalne valute Bitcoin. Potrebno je poudariti, da te zlorabe niso omejene na grafično okolje, kot so pokazali v [3], kjer so avtorji naredili beležnik tipkanja (angl. Keylogger), ki teče samo na grafični kartici. V tem članku avtorji izkoriščajo eno izmed lastnosti grafičnih kartic, in sicer to, da lahko grafična kartica sama dostopa direktno do pomnilnika brez posredovanja centralne procesne enote. To ji omogoča, da prebere vsebino tipkovničinega medpomnilnika. Vse to lahko naredi brez posredovanja procesorja in brez, da bi spreminjala dele kode jedra ali podatkovnih struktur.

Največji problem pri zlonamerni kodi, ki se izvaja na grafični kartici je ta, da jo je zelo težko zaznati. Orodja za digitalno forenziko imajo danes nepopoln pregled nad spominom, saj imajo običajno dostop le do spomina operacijskega sistema (npr. orodji pmem in LiME). Ker pa se ta zlonamerna koda skriva na pomnilniku grafične kartice, jih večina orodij ne

more zaznati. Seveda pa je za tako skritost potrebna čista odcepitev zlonamerne kode od centralne procesne enote. Ravno to pa je ena izmed glavnih nejasnostih, ko govorimo o zlonamerni kodi, ki teče na grafični kartici. Ker na tem področju še ni bilo izvedeno veliko raziskav ni popolnoma jasno do kake mere se zlonamerna koda lahko izvaja brez vednosti glavne procesne enote. Zato tudi ne vemo ali je tradicionalna tehnika digitalne forenzike, kjer najprej pomnilniške podatke zberemo in jih nato analiziramo, sploh dovolj, da identificiramo zlonamerno kodo in jo tudi pregledamo.

3. TEHNIČNO OZADJE

V tem poglavju opišemo sodobno arhitekturo grafičnih kartic, delitev naslovnega prostora med sistemski pomnilnik in pomnilniško preslikane vhodno-izhodne naprave ter sodobna orodja za zajem vsebine pomnilnika v forenzični preiskavi.

3.1 Arhitektura grafičnih kartic

Ločitev centralne procesne enote za splošna procesna opravila in grafične kartice za paralelna opravila je dandanes najbolj običajen primer heterogene računalniške arhitekture. Sodobne grafične kartice so za doseganje učinkovitosti procesiranja podatkovno-paralelnih problemov zgrajene povsem drugače kot običajne centralne procesne enote. Medtem ko je večina vezja na grafičnih karticah namenjena procesiranju podatkov, je na centralni procesni enoti večina prostora namenjena predpomnilniku in kontrolni enoti. Večina današnjih računalnikov je zasnovanih tako, da je grafična kartica preko vodila PCI povezana v sistem. Nekateri sodobni procesorji imajo grafične enote že vgrajene v čip, kar občutno zmanjša latenco pri komunikaciji, poleg tega pa lahko take grafične enote uporabljajo kar glavni pomnilnik namesto svojega lastnega.

Interno so grafične kartice zgrajene hierarhično. Procesorski elementi, ki izvajajo računske operacije, so združeni v bloke, ki vsebujejo tudi nekaj lastnega pomnilnika. Operacije na blokih tečejo sinhrono v načinu SIMT (Single Instruction Multiple Thread), torej v določenem trenutku vse enote v bloku izvajajo isto operacijo. Sami bloki običajno delujejo asinhrono, obstajajo pa tudi sinhronizacijski mehanizmi. Med bloki je običajno rezdeljen še en nivo pomnilnika, ki lahko deluje kot predpomnilnik pred glavnim grafičnim pomnilnikom.

Na grafični kartici je poleg računskega vezja tudi krmilnik, s katerim komunicira gonilnik, ki teče na centralnem procesorju. Gonilnik poleg pravilne komunikacije z grafično kartico kot vhodno-izhodno napravo skrbi tudi za sledenje stanja naprave in vseh povezanih procesov. Uporabniki torej do storitev grafične kartice dostopajo prek uporabniškega procesa, ki prek programskega vmesnika (npr. OpenGL, OpenCL, CUDA ...) dostopa do gonilnika. Uporabniki programe za grafično kartico običajno pišejo v visokonivojskem jeziku, ki ga gonilnik prevaja preden ga pošlje na grafično kartico za izvedbo. Programi, ki tečejo na grafični kartici, se imenujejo *jedra*. Jedra nato kontrolna enota grafične kartice po ukazu za izvedbo opravila razporedi na bloke skupaj z metapodatki, kot so npr. dimenzija opravila, identifikacijska številka jedra ipd.

3.2 Naslovni prostor

Računalniški sistem za naslavljanje podatkov uporablja naslovni prostor, ki je običajno razdeljen v več ločenih delov, npr. sistemski pomnilnik in pomnilniško preslikan vhod in izhod. Iz vhodno-izhodne naprave je naslovni prostor viden precej drugače, saj morebitno preslikavo med logičnim in fizičnim naslovom izvaja ločeno od centralne procesne enote. To predstavlja velik problem, saj na vhodno-izhodnih napravah običajno nimamo takih varnostnih mehanizmov kot na centralni procesni enoti, npr. varovanje pomnilniških dostopov, dovoljenja za dostop do pomnilniških strani ipd. Posledica tega je, da lahko ob previdnem spreminjanju preslikovalnih tabel dostopamo do poljubnih podatkov v sistemskem pomnilniku. Primer take zlorabe je opisan v poglavju 4. Prav tako ima drugačen pogled na pomnilniški prostor krmilnik pomnilnika, kar seveda povzroča podobne varnostne težave.

3.3 Zajem podatkov v forenzični analizi

Forenzični preiskovalec lahko pridobi podrobne informacije o sistemu in njegovem delovanju z analizo pomnilnika - tako glavnega kot grafičnega. S podrobno preiskavo lahko odkrije tudi sledove zlonamerne kode, ki je tekla oz. še teče na sistemu. Za forenzično preiskavo je torej kritičnega pomena, da lahko pridobimo in shranimo trenutno stanje pomnilnika. Pri tem predpostavljamo, da je program pustil sledove v pomnilniku. Obstaja več orodij, s katerimi lahko zajamemo sliko pomnilnika. V tem delu smo preizkusili dva, ki ju navaja tudi članek [2]: *pmem* in *LiME*. Obe orodji naložita modul jedra (angl. kernel module) in opravilo zajema izvedeta v varnostnem načinu jedra (angl. kernel mode). Glavno težavo pri uporabi teh orodij predstavlja prav različen pogled naprav na naslovni prostor, saj del naslovnega prostora, ki je namenjen vhodno-izhodnim napravam ignorirajo, da ne pride do neželenih stranskih učinkov - branje registrov namreč lahko sproži določeno delovanje vhodno-izhodne naprave, ki lahko bistveno vpliva na izsledke forenzične preiskave ali celo postavi sistem v nestabilno stanje. Ta orodja lahko torej učinkovito uporabljamo le pod predpostavko, da je vse sledi o aktivnosti moč razbrati le iz glavnega pomnilnika.

4. ZLONAMERNA KODA

V tem poglavju raziščemo, če je zajem podatkov iz glavnega pomnilnika dovolj za pridobitev vseh relevantnih podatkov za forenzično analizo ali moramo v vsakem primeru zajeti tudi vsebino grafičnega pomnilnika. Pri tem poskušamo ugotoviti ali je mogoče skriti aktivnosti tako, da ne pustijo sledi v glavnem pomnilniku.

Za dobro razumevanje in boljšo preglednost smo zlonamerno kodo klasificirali v več različnih razredov. Razredi se med seboj razlikujejo po tem, koliko pravic ter koliko informacij o grafični kartici zlonamerna koda za svoje delovanje potrebuje. Čeprav so si kategorije med seboj zelo podobne, je s stališča forenzične preiskave lahko razlika med preiskovanjem ogromna.

V prvo kategorijo smo umestili najpreprostejšo vendar v nekaterih primerih tudi najučinkovitejšo zlonamerno kodo. To je koda, ki za svoje izvajanje ne potrebuje posebnih administratorskih pravic in jo lahko izvaja navadni uporabnik. Ta vrsta zlonamerna kode izrablja zmožnost splošnega uporabnika, da lahko direktno upravlja z grafično procesno enoto.

Za svoje delovanje niti ne izrablja nobenih pomanjkljivosti ali ranljivosti grafičnih kartic, pač pa uporablja le splošne programabilne vmesnike, ki so uporabniku vedno dostopni.

V naslednjo kategorijo uvrščamo kodo, ki za svoje izvajanje potrebuje pravice priviligiranega uporabnika. Ta kategorija je po načinu zlorabe zelo podobna prejšnji, saj prav tako za svoje izvajanje ne izrablja ranljivosti ali pomanjkljivosti. Razlikujeta se po tem, da zlonamerna koda v tej kategoriji potrebuje administratorske pravice za svoje izvajanje.

Kategorija, ki jo smatramo za najnaprednejšo, prav tako za svoje izvajanje potrebuje administratorske pravice. Poleg tega pa za razliko od prejšnjih dveh pozna sestavo grafičnega gonilnika. To omogoča, da zlonamerna koda izvaja direktno upravljanje z objekti jedra (angl. Direct Kernel Object Manipulation, DKOM), kar ji omogoča še večji nadzor nad našo grafično procesno enoto.

Glede na te kategorije mora digitalni forenzik nato prilagoditi svojo preiskavo, saj ima vsaka kategorija svoje značilnosti na katere je potrebno biti pozoren. Te kategorije smo v naši raziskavi uporabili za predstavitev rezultatov, saj nas je zanimalo kakšne pravice potrebuje posamezna metoda izrabe grafične procesne enote za svoje delovanje.

Grafični gonilnik skupaj z operacijskim sistemom običajno hrani podatke o tem katera opravila so trenutno v izvajanju in katera so za izvajanje predvidena.Ker se večina teh podatkov nahaja v sistemskem pomnilniku, se zdi, da bi lahko z dobro zasnovanim forenzičnim orodjem preiskali le sistemski pomnilnik in tam našli podatke o vseh opravilih. Pokazali bomo, da je v večini primerov res tako, vendar pa je treba biti pri preiskovanju zelo previden, saj obstajajo tehnike, ki omogočajo, da se opravilo izvaja brez vednosti procesorja. V nadaljevanju predstavimo štiri različne tehnike, ki jih pisci zlonamerne kode lahko uporabijo pri pisanju zlonamerne kode, ki za svoje izvajanje uporablja grafično procesno enoto.

1. **"Neskončno" izvajanje kode** - Ponavadi je za izvajanje programa na grafični kartici potreben nek kontrolni proces, ki teče na gostitelju. Gostitelj doda proces v vrsto iz katere potem grafična procesna enota pridobiva opravila in jih izvaja. Grafična procesna enota opravila izvaja tako, da se med izvajanjem zaklene in opravilo izvaja dokler se ne konča. To lahko privede do mnogih problemov, kot npr. neodziven uporabniški vmesnik. Da se to ne bi zgodilo, imajo grafične kartice nastavljeno časovno omejitev po kateri, če se opravilo ni do konca izvedlo, ga ubijejo. To predstavlja večjo omejitev za zlonamerno kodo, saj če hoče čim več časa biti v izvajanju, se mora v neki zanki kar naprej pošiljati na grafično procesno enoto. To pa pomeni, da jo je veliko lažje zaznati z forenzičnimi orodji. Vendar pa tudi ta omejitev ni dovolj za preprečitev take zlonamerne kode, saj ima grafična kartica to omejitev določeno. Če ima zlonamerna koda pravice, da to omejitev spremeni, je možno, da prevzame nadzor nad grafično procesno enoto in se izvaja dlje časa.

2. **Uporaba GPU brez sodelovanja CPU** - Izvedba opravila na grafični procesni enoti običajno potrebuje prisotnost nekega procesa na gostitelju. To je za forenzično analizo zelo pomembno dejstvo, saj prek procesa na gostitelju lahko odkrijemo katero opravilo se na grafični procesni enoti izvaja. Kot pa smo pokazali, je možno, da povezavo med njima prekinemo in kot rezultat dobimo opravilo na grafični procesni enoti, ki ni povezano z nobenim procesom na gostitelju. To je iz stališča forenzične preiskave zelo zaskrbljujoče, saj je identifikacija take zlonamerne kode mnogo zahtevnejša. Tako kodo je še vedno mogoče odkriti, če forenzik pogleda gonilnik grafične procesne enote, ki se nahaja v gostiteljevem pomnilniku. Tak pregled razkrije prisotnost konteksta grafične procesne enote, ki pa ga ne upravlja noben proces.

3. **Uporaba GPU brez konteksta** - Pri prejšnji tehniki smo omenili, da se opravilo, ki se izvaja na grafični procesni enoti, kljub temu, da nima nobenega povezanega procesa nagostitelju, da prepoznati s pomočjo konteksta na gonilniku grafične procesne enote. Bolj napredna tehnika pa omogoča tudi ločitev opravilovega konteksta in izbris vseh njegovih sledi iz gonilnika. Vse to je seveda mogoče samo, če je program že spremenil časovno omejitev izvajanja. Kot smo pokazali je tudi ta vrsta zlonamerne kode možna in jo je zelo zahtevno zaznati.

4. **Nekonsistentno pomnilniško preslikovanje** - Tako operacijski sistem kot grafična kartica imata vsaka svoj pomnilniški prostor, ki je med seboj preslikan. Kot smo pokazali je možno, da se informacije vsebovane v preslikanih tabelah ne ujemajo. S tem je možno doseči zlonamerno kodo, kot je beležnik tipkanja, ki deluje izključno na grafični kartici in do podatkov dostopa preko neposrednega dostopa do pomnilnika (angl. Direct memory access).

5. ŠTUDIJA PRIMERA

Navedene trditve in hipoteze smo preverili na računalniškem sistemu z operacijskim sistemom Linux in Intelovo integrirano rešitvijo Intel HD Graphics. Po trditvah avtorjev članka [2] naj bi bili koncepti in ideje na različnih platformah primerljivi. Operacijski sistem Linux storitve grafične kartice ponuja prek ogrodja *Direct Rendering Infrastructure* (DRI). V tem ogrodju lahko uporabniški programi komunicirajo z grafično kartico neposredno prek sistemskega klica *ioctl* (input-output control) ali posredno prek določenega grafičnega ogrodja, npr. X11 ali Wayland.

Celotno ogrodje DRI je razdeljeno na tri komponente:

1. **CRT krmilnik (CRTC)** - modul jedra, ki skrbi za komunikacijo s krmilnikom zaslona.

2. **Graphics Execution Manager (GEM) oz. Translation Table Maps (TTM)** - modul jedra, ki skrbi za nadzor grafičnega pomnilnika.

3. **Gonilnik naprav**, ki visokonivojske ukaze programskih vmesnikov (npr. OpenGL ali OpenCL) prevaja v nizkonivojske ukaze, specifične za določeno strojno opremo.

Prvi dve komponenti skupaj tvorita *Direct Rendering Manager* (DRM), podsistem jedra operacijskega sistema, ki vsebuje celotno funkcionalnost za dostop do informacij sistema, stanja in fizičnih lastnosti naprav (npr. priključki), pa tudi enoten uporabniški programski vmesnik, s katerim lahko aplikacije upravljajo z grafičnim pomnilnikom, prekinitvami, DMA prenosi itd. Podsistem DRM vključuje tudi komponento *libdrm*, ki teče v uporabniškem načinu in služi kot uporabniško dostopen vmesnik do modulov jedra. Knjižnica libdrm je množica ovojnih funkcij do sistemskega klica ioctl in je specifična za določeno napravo. Proizvajalci naprav skupaj s svojimi gonilniki nudijo še knjižnico libdrm, ki specifike naprave skrije pred uporabnikom in višjim slojem v arhitekturi DRI nudi enoten uporabniški programski vmesnik.

DRM podpira dva upravljalnika s pomnilnikom: Translation Table Maps (TTM) in Graphics Execution Manager (GEM). Intelov gonilnik uporablja GEM, zato se tudi v tem besedilu osredotočamo na ta upravljalnik. GEM je v svojem bistvu le generičen programski vmesnik, gonilniki za specifične naprave pa ta vmesnik implementirajo.

V našem primeru je bil uporabljen uporabniški programski vmesnik Beignet, odprtokodna implementacija standarda OpenCL. Beignet komunicira s knjižnico libdrm in tako dostopa do storitev strojne opreme. OpenCL (Open Computing Language) je odprtokodni standard za programiranje paralelnih naprav, kot so npr. grafične kartice, digitalni signalni procesorji in FPGA naprave. Beignet OpenCL je sestavljen iz dveh delov: modula za izvajanje in prevajalnika. Prevajalnik prevaja kodo, napisano v jeziku OpenCL C, v vmesno kodo LLVM, ki se nato prevede še v strojno kodo za specifično napravo. Sprotno prevajanje (angl. Just-In-Time, JIT) je v takem sistemu neizbežno, saj moramo zagotoviti delovanje kode na različnih platformah. Poganjanje strojne kode in upravljanje naprave poteka prek knjižnice libdrm, ki ga nadzoruje Beignetov modul za izvajanje opravil.

Funkcionalnosti, specifične za Intelove grafične naprave, implementira Intelov gonilnik i915 v Linuxovem DRM ogrodju. Gonilnik komunicira neposredno z napravo prek pomnilniško preslikanih registrov ali prek seznamov ukazov. Teh seznamov je več za različne vrste opravil (npr. upodabljanje in podatkovne prenose). Gonilnik poleg komunikacije skrbi tudi za ustvarjanje kontekstov, prek katerih upravlja tudi s stanjem naprave. Na novejših napravah obstajajo tudi strojni konteksti, ki efektivno razbremenijo gonilnik in omejijo možne napade nanj. V primeru uporabe strojnih kontekstov gonilnik naprave le pošilja ukaze za menjavo kontekstov, vsa ostala opravila pa se izvajajo znotraj naprave brez posredovanja gonilnika. Vsi strojni konteksti so v gonilniku shranjeni v povezanem seznamu struktur tipa `i915_hw_context`.

V naši raziskavi je bistvenega pomena razumevanje različnih pogledov naprav na naslovni prostor. Centralna procesna enota naslovni prostor vidi povsem drugače kot krmilnik pomnilnika ali grafična kartica. Nekateri naslovi so namreč namenjeni pomnilniško preslikanim vhodno-izhodnim napravam in vodilu PCI, medtem ko krmilnik pomnilnika teh naslovov nima ločenih, ampak na prostor gleda le kot seznam enakovrednih zaporednih naslovov. Razdelitev naslovnega prostora (angl. address space layout) na procesorjih z arhitekturo Haswell je prikazana na sliki 1. S slike je razviden tudi prostor, do katerega centralna procesna enota ne more dostopati, imenovan *ukradeni prostor* (angl. Graphics Stolen Memory, GSM). Sestavljata ga dve območji: pomnilniške preslikovalne tabele (angl. Graphics Translation Tables, GTT) in podatkovni prostor (npr. slikovni medpomnilnik). Bazni register preslikovalnih tabel (BGSM) in register, ki označuje konec nižjega naslovnega prostora (TOLUD), sta označena na sliki 1. Centralna procesna enota lahko dostopa do ukradenega prostora prek pomnilniško preslikanega vhoda in izhoda prek baznih registrov vodila PCI št. 0 in 2 (BAR0 in BAR2). Grafična kartica uporablja dve različni preslikovalni tabeli, shranjeni lokalno, za preslikavo svojih lastnih virtualnih naslovov v fizične naslove. Prva tabela je globalna (angl. Global Graphics Translation Table, GGTT) in slika v prostor globalno dostopnih podatkov (npr. slikovni medpomnilnik in globalne funkcije za upodabljanje), druga pa je lokalna za vsak proces (angl. Per-Process Graphics Translation Table, PPGTT). Globalna tabela lahko slika tako v ukradeni prostor kot v sistemski pomnilnik, medtem ko lahko lokalna tabela procesa slika le v sistemski pomnilnik. Vnosi, ki se ne preslikajo v GSM, so enaki v obeh tabelah.

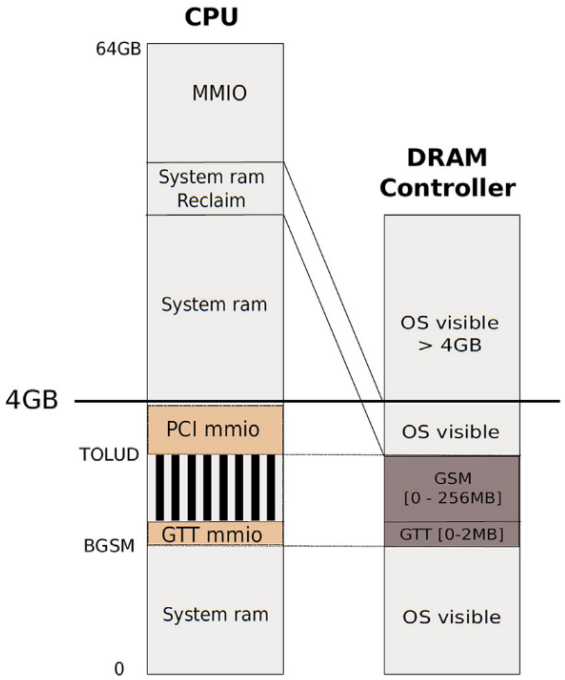


Figure 1: Razdelitev naslovnega prostora na procesorjih z arhitekturo Haswell.

5.1 Empirična raziskava

V tem poglavju bomo predstavili rezultate empirične raziskave, ki smo jo izvedli na prej opisanem sistemu. Rezultate raziskave predstavimo glede na štiri kategorije zlonamerne kode, ki smo jih opisali v poglavju 4.

1. **"Neskončno" izvajanje kode** - Da zlonamerna koda

doseže ”neskončno”izvajanje kode, mora v Intelovem gonilniku onemogočiti *hangcheck*. Hangcheck je mehanizem, ki ubije vsako opravilo, ki se na grafični kartici izvaja več časa, kot je zapisano. Pri Intelovem i915 gonilniku je možno hangcheck onemogočiti prek path/sys/module/i915/parameters/enable _hangcheck s tem da v datoteko zapišemo ničlo.Ta zlonamerna koda spada v kategorijo, ki za svojo operiranje potrebuje posebne administratorske pravice.

2. **Uporaba GPU brez sodelovanja CPU** - Pokažemo, da je možno izvajanje opravila na grafični procesni enoti brez procesa na gostitelju. V eksperimentu je naša zlonamerna koda izvajala neskončno zanko, ki je le povečevala vrednost števca na skladu gostitelja. Takoj, ko je grafična procesna enota začela izvajati našo kodo, smo s pošiljanjem signala SIGINT naši zlonamerni kodi, ubili njen proces na gostitelju. Da smo preverili ali se naša koda še vedno izvaja smo uporabili več indikatorjev. Najprej smo ocenili obremenitev grafične kartice, ki je pokazala, da se na grafični procesni enoti izvaja neko opravilo. Pogledali pa smo tudi vrednost, ki jo naša zlonamerna koda na gostitelju spreminja in ugotovili, da se ta vrednost ves čas spreminja. Na gostitelju pa se procesa, ki bi povezoval opravilo na GPUju ni videlo. Ta zloraba pride v poštev samo če zlonamerna koda prej prav tako pride v stanje ”neskončnega”izvajanja. Vendar pa bi tak napad povzročil neodzivnost grafičnega vmesnika na gostitelju in bi se zelo hitro opazil.

3. **Uporaba GPU brez konteksta** - Pokažemo, da je možno izvajanje zlonamerne kode, brez da bi se v gonilniku grafične kartice videl kontekst opravila. Ponovili smo enak eksperiment kot pri prejšnjem primeru vendar pa tokrat nočemo, da se vidi kontekst opravila. Za izvedbo te metode poleg administratorskih pravic potrebujemo tudi detajlno znanje o gonilniku grafične kartice, da zlonamerna koda lahko izvaja direktno upravljanje z objekti jedra. Da smo dosegli odstranitev konteksta, smo naredili še eno zlonamerno opravilo, ki locira kontekst prejšnjega opravila in odstrani njegovo referenco iz notranjega seznama gonilnika. Najprej je potrebno najti simbol, ki referencira strukturo `drm_i915_private`. S pomočjo te strukture pridobi kazalec na lokacijo, kjer se nahaja seznam kontekstov. Nato uniči vse te kontekste s pomočjo funkcije `i915_gem_context_unreference()`. Ta funkcija ne uniči samo našega konteksta, pač pa uniči kontekste vseh opravil, kar je za forenzično analizo dobra lastnost, saj je to dejavnost precej lahko zaznati. Težje pa je, če je zlonamerna koda edino opravilo, ki se izvaja na grafični procesni enoti, saj ta izbris potem veliko težje zaznamo. Poleg tega pa se izvajanje kode z izbrisom konteksta ne spremeni.

4. **Nekonsistentno pomnilniško preslikovanje** - Grafična procesna enota in centralna procesna enota uporabljata za objekt, ki ga alocira grafična kartica različna virtualna naslova. Ob normalnem izvajanju oba naslova kažeta na enak pomnilniški naslov. Vendar pa je preslikovanje centralne procesorske enote shranjeno v tabelah strani operacijskega sistema, grafične procesne enote pa tabelah strani grafične kartice. Torej, če nam uspe spremeniti naslov v tabeli strani grafične kartice, ostane naslov na operacijskem sistemu nespremenjen.

   Pri izvedbi eksperimenta smo se osredotočili na pridobivanje podatkov, ki jih shranjuje aplikacija, ki za svoje delovanje ne uporablja grafične kartice. S svojo aplikacijo najprej izvedemo alokacijo svojega medpomnilnika velikosti 1024MB. Nato izvedemo kodo, ki najde pomnilniški naslov aplikacije in našega alociranega medpomnilnika. Nato pridobi kazalec na naslov začetka področja GSM preko strukture `drm_i915_private`. Nato pregleda dva nivoja PPGTT in najde zapis, ki kaže na fizični naslov našega medpomnilnika in ga zamenjamo z naslovom aplikacije katere podatke želimo dobiti. Nato pripravimo opravilo, ki prepiše vsebino našega medpomnilnika, ki pa v resnici kaže na naslov napadene aplikacije. S pomočjo te tehnike smo dosegli zlonamerno kodo, ki uporablja nekonsistentno pomnilniško preslikovanje. Digitalni forenzik bi pri preiskavi sistemskega pomnilnika videl, da naše opravilo lahko dostopa le do našega alociranega medpomnilnika, čeprav v resnici kaže na naslov napadene aplikacije, saj je prišlo do razlike med tabelami strani CPU in GPU.

Anti-forensic Technique	Malware Requirements	Forensic objectives: List processes	List kernels	Memory map
None	U	✓ (OS)	✓ (Driver)	✓ (OS)
Unlimited Execution	S	✓ (OS)	✓ (Driver)	✓ (OS)
Process-less	S	✗	✓ (Driver)	✓ (Driver)
Inconsistent Map	K	✓ (OS)	✓ (Driver)	✗
Context-less	K	✗	✗	✗

Figure 2: Rezultati naše raziskave.

Na sliki 2 smo združili rezultate glede na različne kategorije, na katere smo razdelili zlonamerno kodo. V drugem stolpcu so prikazane zahteve zlonamerne kode, ki jih ta potrebuje, U pomeni uporabniške pravice, S pomeni administratorske pravice in K pomeni, da zlonamerna koda potrebuje poznati tudi zgradbo gonilnika. Na desni strani tabele pa povzamemo posledice na digitalno forenziko, razdeljeno na tri glavne skupine izpisovanja procesov, ki uporabljajo GPU. Znak ✓pomeni, da je ta forenzični cilj možno doseči z določeno tehniko digitalne forenzike. To je predvsem pomembno zato, ker to pomeni, da imamo za te preglede že razvita forenzična orodja in jih ni potrebno prilagajati. Bolj zaskbljujoči so postopki, ki so na sliki označeni s ✗, saj za njih ne obstajajo forenzična orodja, s katerimi bi te zlorabe lahko odkrili. Potrebno bi bilo razviti posebna orodja za digitalne preiskave, s katerimi bi lahko preiskali tudi pomnilnik grafične procesne enote in tam poiskali zlonamerno kodo, kar pa bi znal biti problem, saj so specifikacije za grafične kartice in njihove gonilnike precej nepoznane, saj jih proizvajalci ne razkrivajo javnosti.

6. SKLEPI IN NADALJNJE DELO

V tem članku smo opisali nekatere možnosti uporabe grafičnih kartic kot orodja za izvajanje zlonamernih aktivno-

sti. Na primeru Intel HD Graphics smo pokazali različne načine skrivanja aktivnosti in onemogočanja forenzične analize. Uporabili smo sistem Linux z odprtokodno knjižnico Beignet in Intelovim gonilnikom i915. Predstavili smo vse uporabljene tehnologije in kako se le-te vključujejo v grafično ogrodje operacijskega sistema Linux.

Pokazali smo, da je mogoče iz centralne procesne enote dostopati do kritičnih delov gonilnika in strojne opreme ter s tem skrivati zlonamerne aktivnosti. Raziskali smo štiri različne načine zlorabe: "neskončno"izvajanje kode, uporaba GPU brez sodelovanja CPU, uporaba GPU brez konteksta in nekonsistento pomnilniško preslikovanje. Ugotovili smo, da so te zlorabe relativno enostavno izvedljive na Intelovem gonilniku in integrirani grafični kartici, vendar pa njihovo praktično uporabo ovira prisotnost varnostnih mehanizmov operacijskega sistema (predvsem zahteva po administratorskih pravicah). Na sliki 2 so s simbolom ✓označene zlorabe, ki jih z že narejenimi forenzičnimi orodji lahko odkrijemo, medtem ko so ostale neizsledljive.

Ker smo se v tem besedilu omejili le na določena orodja in tehnologije, je prispevek našega članka omejen na ozko množico platform. Opisane ranljivosti so lahko v nekaterih okoljih povsem zavarovane, medtem ko obstaja potencialno mnogo ranljivosti, ki jih nismo opisali. V najslabšem primeru moramo vsako okolje obravnavati ločeno in z ozirom na njihove specifike. Na sistemih Linux so te težave nekoliko omiljene s prisotnostjo enotnih programskih vmesnikov, toda neposreden dostop pogosto ni opcija zaradi lastniške kode, v katero nimamo vpogleda. Tudi pri forenziki pomnilnika smo trenutno omejeni na specifične arhitekture in obstoječa orodja, kajti v najslabšem primeru je potrebno razviti ločena orodja za vsako napravo posebej.

Kljub tem omejitvam ta prispevek nudi vpogled v relativno novo področje računalniške forenzike in odpira veliko možnosti za nadaljnje raziskave.

7. REFERENCES

[1] ArsTechnica. Bitcoin malware uses gpu for mining, 2015.

[2] D. Balzarotti, R. D. Pietro, and A. Villani. The impact of gpu-assisted malware on memory forensics: A case study. *Digital Investigation*, 14, Supplement 1:S16 – S24, 2015. The Proceedings of the Fifteenth Annual {DFRWS} Conference.

[3] E. Ladakis, L. Koromilas, G. Vasiliadis, M. Polychronakis, and S. Ioannidis. You Can Type, but You Can't Hide: A Stealthy GPU-based Keylogger. In *Proceedings of the 6th European Workshop on System Security*. EuroSec, Prague, Czech Republic, April 2013.

Volatile memory forensics and rootkit detection on OS X

[Survey of the field, comparison of tools and an experiment]

Jaka Šušteršič Faculty of Computer and Information Science
Večna pot 113
1000 Ljubljana
js1929@student.uni-lj.si

Gašper Slapničar Faculty of Computer and Information Science
Večna pot 113
1000 Ljubljana
gs6068@student.uni-lj.si

ABSTRACT

The first part of our study reviews the article of A. Case titled *Advancing Mac OS X rootkit detection*, alongside three important related studies in this field. It highlights the increasing importance of both OS X and volatile memory analysis in forensic science as valuable potential evidence. It focuses on rootkits as an increasingly popular yet under-researched form of malware and reviews some rootkit detection techniques. The Volatility framework and its rootkit detection plugins for Mac OS X are discussed in detail. In the second part our study describes an experiment in which we analyzed a RAM dump originating from a sample OS X device. A variety of tools were used to extract sensitive data from the system key chain and an application specific data store. Such data is shown to potentially be of vital importance to a forensic specialist.

Categories and Subject Descriptors

[**Applied computing**]: Computer forensics, System forensics, Investigation techniques

Keywords

Mac OS X, malware, rootkit, kernel, forensics, volatile memory analysis, Random Access Memory (RAM), forensic tools

1. INTRODUCTION

In recent years, there has been little change in the global desktop operating system market. Windows operating system variants account for nearly 82% of the market, while OS X accounts for only 9.5%.[9] Windows' large market share has historically made it a big target for authors of malicious software, while OS X had received much less attention. This paradigm is slowly shifting towards OS X, as it has become the most popular choice of operating system among professional software developers in 2016 [4]. Devices owned by these type of users are very likely to contain credentials for production systems and other sensitive data which makes them valuable targets.

Big advances in malware detection were made in years 2013 and 2014 when nation-state backed malware samples were found and examined [18, 16, 15]. Research in this time span mostly focused on kernel components used by previously detected malware samples, which account for only a fragment of the possible types of attacks. Broader systematic research was already available for operating systems such as Windows and Linux but was not applied on OS X-specific subsystems.

In the first part of the paper we review some studies in the related field and provide an in-depth description of the plugins developed in the study under review.

Our focus is on live forensics and memory analysis which are considered essential when dealing with volatile data, such as advanced rootkits and related data that does not reside in permanent storage and may be lost by powering down a suspects system, such as running processes, open files by processes, usernames and passwords, decrypted versions of a program, current network connections, encryption keys, etc. Using traditional storage forensics during such a triage often results in the most important piece of evidence – physical memory (RAM) being lost [19].

In the second part of the paper we describe an experiment conducted on a sample OS X device in which a RAM dump was analyzed and vital pieces of data were collected using a variety of tools.

2. RELATED WORK

As Mac OS X rootkit detection is a rather recent field of study, most related sources are also recent and were presented in the past five years. In the following sections we review some of the major studies that influenced the development of rootkit detection in Mac OS X.

2.1 Advanced Mac OS X physical memory analysis (Suiche M.) [20]

One of the first studies in this field, laying a foundation for understanding and detection of rootkits on Mac OS X, was presented in 2010. The author highlights the forensic requirement of researching volatile (physical) memory along with the pros (sometimes volatile physical memory data is required) and cons (high complexity) of such memory analysis. It

focuses on processes, kernel extensions, mounted file system, machine information and syscalls as shown in figure 1.

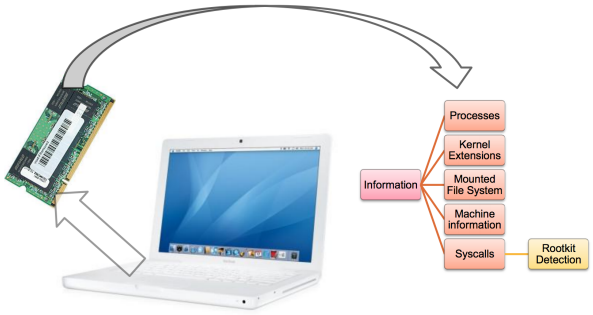


Figure 1: A schematic showing the main highlights of the study given by M. Suiche in 2010 [20].

The ultimate goal is to avoid random string searching of `/dev/mem` and to rather propose a precise and efficient method for anomaly detection. In order to achieve this, the author first discusses the parts of Mac OS X shown in figure 1 in great detail. First step is to compute the kernel physical address (KPA) from the kernel virtual address (KVA) using the simple translation formulas shown in table 1.

Operating System	Quick translation formula
x86 Linux	KPA = KVA – 0xC0000000
x86 Windows	KPA = KVA & 0x1FFFF000
Mac OS X	KPA = KVA

Table 1: Quick translation formulas to obtain KPA from KVA.

A crucial element that makes advanced memory analysis possible are *symbols*. On Mac OS X symbols are directly stored inside the executable in a segment/section called `__LINKEDIT`. It takes some effort to retrieve these symbols. Example dump of retrieved symbols is shown in figure 2.

```
[000223] _IOZeroTvalspec                                    0x0050EE18
[000224] _IS_64BIT_PROCESS                                  0x00373952
[000225] _IdlePDPT                                          0x004EB008
[000226] _IdlePDPT64                                        0x004EB010
[000227] _IdlePML4                                          0x004EB00C
[000228] _IdlePTD                                           0x004EB004
[000229] _InitGlobals                                       0x002A3A08
[000230] _InsertKeyRecord                                   0x003477C8
[000231] _InsertOffset                                      0x003475A1
[000232] _InsertRecord                                      0x00347703
[..]
```

Figure 2: An example dump of symbols on Mac OS X.

Once physical memory manager using symbols is functional, information can be extracted. The process is described in great detail in the article itself [20].

The final proposed method is to check if an offset from a syscall entry is not present in the kernel symbols, and deduct that this signifies an anomaly. This is a rather simple and fast method for detecting anomalies and potential rootkits.

2.2 Mac Memory Analysis with Volatility (Case A.) [10]

In 2012, Mac OS X support was added to Volatility software as presented by the author of this study. The motivation at the time was that good tools for aquisition of memory information from Mac OS X were already available, however no tools for robust deep analysis of the captured memory existed. The author first presents the process of memory capture (e.g. RAM) highlighting the complexity of such data containing the entire state of the operating system along with all running applications and related data structures, variables, etc. The goal of analysing such data is to obtain higher level objects representing in-memory data structures and variables related used by the operating system. With these, process information, file listings and networking data can be recovered.

As modern versions of Mac no longer support `/dev/mem` or `/dev/kmem`, this means that 3rd party software must be used to access physical memory data. Only notable example of such a program at the time (up to Mac OS X 10.9) was *Mac Memory Reader* which was free but not open source. It worked by recreating `/dev/mem` and then captured data from it.

The study continues by suggesting how to recover information about processes, memory maps, open files, network connections, network data, loaded kernel modules and *rootkit detection*. Most commonly a distinction is made between *unpriviliged (userland)* and *fully priviliged (kernel) rootkits*, however Mac OS X blurs this line with its micro-kernel design.

Two types of rootkits are mentioned:

1. **Static.** These alter data that is set at compile time and never changes. Examples: code instructions and modifying system call table entries.
2. **Dynamic.** Alters data that is only created and referenced at runtime. Example: manipulation of live data structures used by the OS for core operations. This type of rootkit is more challenging.

A few rootkit examples are also discussed. *Logkext* is a rootkit that uses IOKit framework to log user keystrokes and gives the rootkit control every time a key is pressed. *IP Filter Rootkits* use IP Filters, which is part of the Network Kernel Extension framework and it allows the programmer to easily hook incoming/outgoing packets. Further more, these hooks can modify packets in-place which allows for obvious abuse.

Support for detection of such rootkits was added by the author to the Volatility software, enabling it to be successfully used by forensic investigators on Mac OS X.

2.3 Forensic Investigation into Mac OS X Volatile Memory (Coppock T. & Gan D.) [13]

A rather recent 2014 study which reviewes the current state of live memory analysis tools for Mac OS X and proposes

a new tool called VolaGUI to assist forensic examiners in analyzing volatile memory.

The authors of this study first highlight the increasing importance of digital evidence on court and suggest that physical memory should never be overlooked as it can provide vital and substantial information from a RAM dump. However this is noted to be difficult, especially since ACPO (Association of Chief Police Officers) guidelines state that any tool used to acquire a RAM dump must not leave a footprint on the target machine.

A brief overview of three main live memory analysis tools available for Mac OS X is given. All the tools are noted to be challenging to use but of varied complexity and capabilities:

1. **Volatility.** An open source framework written in Python and implemented under GNU GPL (GNU General Public License). It can be used cross platform on Windows, Linux and Mac OS X. It is backed by a large developer and contributer base, however Mac support was only added in version 2.3 as mentioned in the previous study (Case A.). It comes with all the necessary tools for live memory analysis.

2. **Volafox.** An open source GNU GPL command line based tool, built on Volatility memory analysis framework. It is written in Python, making it cross platform. It is based on the idea of symbol table regeneration which is presented into more detail by the first mentioned study (Suiche M.). In its early stages it was heavily reliant on the presence of the kernel image which was deemed inefficient. Later revisions provided an overlay generator for the RAM dump which allows for successful symbol generation.

3. **Goldfish.** Closed source tool available to law enforcement only. As such it has very little available documentation and not much is widely known. It provides a tool to obtain a RAM image which can then be used to recover the system password and AIM (AOL Instant Messenger) chat fragments.

The authors created a GUI for using the Volafox tool without command line interface which simplifies the usage but is not the focus of the topic of our paper.

3. ROOTKIT DETECTION TOOLS

Prior to 2004, memory forensics was done on an ad-hoc basis, using generic data analysis tools like strings and grep which are not specifically created for memory forensics, and are therefore difficult to use [14]. These tools allow for searching of printable characters directly on a live system either by mounting a block device (e.g. `/dev/mem`) which is an image of the main memory of the computer on Linux operating systems, or by using the *PhysicalMemory* device object on Windows variants [6]. Usage of these tools is difficult because little context can be extracted from the resulting data.

New memory analysis tools intended for practical use have recently been developed such as FACE [12] and Volatility. These allow us to understand fragments of extracted data more easily and piece them together to form a bigger picture with much more context than before.

3.1 Volatility Framework

The Volatility Framework is an open source collection of tools, implemented in Python under the GNU General Public License, for the extraction of digital artifacts from volatile memory (RAM) samples. This framework was already discussed briefly in chapter 2.

In the following subsections we focus on Volatility Framework plugins and related OS X subsystems which are demonstrated in the study under review [12]. Each subsection includes the motivation behind the usage of such subsystems and detection techniques.

3.2 Plugins

The Volatility framework is designed to be extended using separate components named plugins. The framework itself provides the building blocks for interacting with the memory samples. These can be used by developers to create operating system and domain specific functionality. The current distribution of Volatility includes around 250 plugins, about 70 of which are OS X specific. The functionalities provided range from simply listing running processes, to finding application-specific encryption keys.

3.2.1 Power event notifications

Windows operating system variants expose power event notifications to kernel modules, which allow them to be notified of power events such as system reboot, shutdown, crash and sleep. This functionality was meant to provide ample time for each kernel component to free resources, flush buffers and set hardware devices to a known state. These notifications can be exploited by malware to ensure MBR-based (Master Boot Record) persistance [5], avoid crash dump detection [17] or to prevent security tools from loading.

In OS X such power event notifications are provided by IOKit. It provides an API for services to register for specific notifications using an *interest request*. These interests are stored in the IOService I/O Registry plane, which consists of multiple planes. Each plane further consists of trees of IORegistryEntry structures.

To detect malware using such event notifications a Volatility plugin named **mac_interest_headers** was created. It traverses the trees in the registry planes and finds nodes which contain power related interests. Each enumerated node handler is then marked suspicious if its address is not within the code section of the running kernel or within the address space of a loaded kernel extension.

3.2.2 Kernel timers

Timers can be used by kernel component to execute functions after a given time period. Legitimate uses include polling of devices and queue clearing. This functionality can be abused by malware to check persistence mechanism status, contact command, control servers, etc.

In OS X extensions can register a function to be called after

a certain period of time has passed using IODelay and IOSleep. These timers are located in rtclock_timer structures, which store a queue of CPU specific timers.

To detect set timers a Volatility plugin named **mac_timers** was created. It first obtains the number of active processors and then collects each processors cpu_data structure. The rtclock_timer member is then examined and deemed possibly malicious based on the kernel module and the address of the handler function.

3.2.3 Driver communication with devfs and IOKit

Drivers that wish to communicate with userland components must set up handlers for the operations that they wish to provide. These include reading, writing, memory mapping, close operations, etc. These are stored as function pointers, which are called when a userland componenet makes a request to the driver.

Malware can overwrite multiple types of handlers to prevent detection, removal and aid in persistence. By changing specific read handlers, malware can filter file data returned to the userland components. This data includes file content, directory listings and even network packets. Changing the write handlers can be used to prevent removal by a security tool or simply disabling it.

OS X allows components to communicate with drivers through either devfs or IOKit. Devfs exists under both Linux and OS X and allows the creation of files under the /dev directory. To detect modified handlers in devfs a Volatility plugin **mac_devfs** was created. It enumerates all block devices and validates their handlers by cross-matching them with the global cdevsw array that holds all the operation pointers.

The IOKit API provides a IOServiceOpen function which returns a handler pointer in a similar way as devfs. In order to detect modified handlers in IOKit a Volatility plugin **mac_kernel_classes** was created. It checks all loaded IOKit C++ classes and verifies each handler based on its code location either inside or outside a known kernel component.

3.2.4 File system hooks (mac_check_fop)

Different file systems on Linux and OS X use isolated implementations of specific file operations. These include support for reading and writing from files, getting a directory's content and obtaining file meta-data. The kernel itself is isolated from the specifics of each file system by using the virtual file system (VFS). Similar to devfs, modification of these function pointers can be of use to malware. A common technique used by malware on Linux is to hook the read directory function of the `/proc` folder to prevent detection of a running process by tools such as ps. A similar effect can be achieved by hooking the read function of the `/proc/net/tcp` folder to prevent detection of network connections.

On OS X malware can hook the VFS operation pointers in 2 locations. The first one is the configuration table for a particular file system. The second one is a file-system specific data structure vnode, which contains all the operations that componenets can request from the file systems.

To detect changes to the file system handlers a Volatility plugin named **mac_check_fop** was created. It first checks all vfstable file-system specific structures and determines if a given operation's function pointer is suspicious. The second step includes checking all vnode operation structures in a similar way.

3.2.5 File system events (mac_vfsevents)

File system events allow components to be notified about events such as creation, deletion and modification. This can be used by malware to monitor security tools and aid persistence.

OS X provides a character device `/dev/fsevents` for interaction with userland components. Processes can register for event watchers, which are stored in the watcher_table global array. A Volatility plugin named **mac_vfsevents** was created to detect such malicious entries. It finds the watcher_table global array and outputs all the processes with watchers. Further process identification can be done with existing Volatility plugins such as mac_procdump.

3.2.6 Events (mac_kevents)

Another interface which provides file system events to userland components in OS X is kqueue. Aside from file system events it also enables notifications about file descriptor events, process events and even signals sent to a process. Using kqueue it is possible to detect when a socket is being read from, when a new connection is initiated and when a process is terminated or forked. These functions can be abused by malware to prevent processes from loading, detect file tampering and aid persistence by being triggered after a certain process has been terminated. A Volatility plugin named **mac_kevents** was created to aid in detection of such malicious activity. It collects all registered file, socket and process entries from all processes and outputs them in a simple manner. These records must be then manually checked for malicious entries.

4. EXPERIMENTS

Replication of an experiment showing rootkit detection on Mac OS X had proven to be difficult due to lack of suitable sample memory data, namely access to a rootkit infected device. The plugins from the study in review were also unavailable. Due to these limitations we focused our experimental efforts on the broader topic of volatile memory analysis on OS X.

4.1 Data acquisition

In order to analyze volatile memory, a data sample had to be acquired. As our resources were limited, we used our own workstations. We chose not to infect them with rootkits but rather obtain a real world sample of volatile memory originating from a daily used computer.

Several memory extraction tools were surveyed, but only MacPmem (a part of the Rekall [8] forensic framework) offered support for the latest OS X operating system version 10.11.4. Before an extraction can start, a kernel extension must be added, which requires the users password. This requirement can be avoided by using a privilege escalation exploit on the device or by using specialized hardware tools.

Hardware tools also have the advantage over conventional software tools that they can extract RAM even if the device is locked. The process of writing the memory data to a file using MacPmem took about 5 minutes on a modern notebook with a solid state drive (SSD).

4.2 Data analysis

The Volatility framework was our primary tool during this phase. The framework requires each loaded memory sample to be associated with a profile file first so it knows which data structures, algorithms, and symbols to use during analysis. Once a correct operating system version specific profile is selected a user can execute a specific plugin.

We first used the mac_pslist plugin to obtain the list of running processes' names and identifiers. The only notable processes found were Chrome and Dropbox. In the next step we used the mac_netstat plugin to view all active network connections, which confirmed that the two applications were active.

An important area of RAM on OS X systems is the file cache region. It contains recently used files from both system and user applications. We were able to obtain a list of stored files and correspondent memory addresses using the mac_list_files plugin. These files could then be saved to the host computer using the mac_dump_file plugin. The combination of these two plugins was vital to our experiment. By manually checking the list of files stored in the file cache we identified and exported two relevant files - login.keychain which contains the OS X keychain data store and Login Data which includes website usernames and passwords stored by Chrome.

The keychain file itself is encrypted using a Master Key, so access to the file as such does not provide usable data. Fortunately the Master Key is also stored in RAM and the Volatility framework includes a plugin named mac_keychaindump which was used to find potential Master Key candidates. Each of the key candidates was then used to try to decrypt the keychain file using a python script called chainbreaker [1]. Once a correct key was found, the script extracted all the data stored in the keychain. A manual survey of the data revealed WiFi 802.11.x passwords, certificates, private keys and application specific data. An example of a WiFi password entry is shown in figure 3.

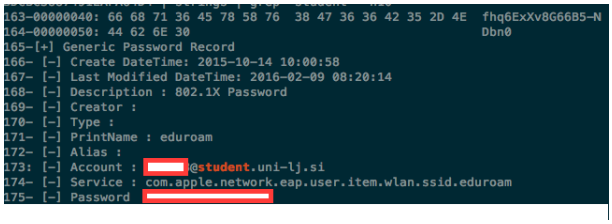


Figure 3: An example of a WiFi password entry in the system keychain.

Our final objective was to obtain the usernames and passwords from the Chrome data store. The Login Data file itself is a SQLite database file. Opening the file in DB Browser for SQLite [3] reveals the database schema. Each record includes the website address, username in cleartext and an encrypted field which contains the password. Using the source code of the Chromium project [2], we were able to determine that the password is encrypted using AES (Advanced Encryption Standard) in CBC (Cipher Block Chaining) mode using a known salt, a platform dependant key and a known number of iterations. The platform dependant key used on OS X distributions of Chrome is named Chrome Safe Storage and was found in the decrypted keychain. Password decryption was then achieved using a modified python script named pycookiecheat [7], which was originally developed for extracting encrypted cookie data. The script connects to the SQLite file and queries it for pairs of usernames and passwords. Each password is then decrypted by using the known salt and Chrome Safe Storage key in 1003 iterations.

4.3 Findings

The experiment has shown that in case of access to a live computer we can rather easily obtain a RAM dump, allowing us to access information that would otherwise be inaccessible after the device was shut down or the user logged out. The Volatility framework includes many plugins which reduce the forensic specialists' workload and provide meaningful data in just a few simple steps.

In our experiment we managed to obtain access to the cleartext of the system keychain and view multiple passwords, certificates and private keys stored on the computer as shown in figure 3. Such information allows the forensic specialist to potentially access remote services and retrieve system encryption keys, which could lead to additional discovery of evidence.

5. CONCLUSION

Our study reviewed some of the approaches to Mac OS X rootkit detection and volatile memory analysis as given by A. Case in his study *Advancing Mac OS X rootkit detection* [11].

The study and related studies have shown that Mac OS X is becoming increasingly popular among software developers and therefore a more attractive malware target. The broader field of volatile memory forensics which analyzes memory on live devices is also gaining importance as such data can provide important information or evidence to forensic specialists.

We reviewed some of the available tools for volatile memory analysis and rootkit detection focusing on the Volatility framework and its OS X specific plugins which were developed recently. Using these (and some other) tools we analyzed a RAM dump sample obtained from a device running OS X and extracted some critical credentials.

Merging the findings and emphases of the study under review and some of the related work we placed rootkit detection on Mac OS X within the broader field of volatile memory analysis and created a condensed and focused summary of these studies. This can benefit both readers trying to deepen their knowledge of the field and experienced forensic specialists trying to apply volatile memory analysis in

real life cases.

The descriptions of Volatility plugins for rootkit detection on OS X given in the study allow for detailed understanding of how certain techniques used by rootkits work, as well as how we can use certain operating system mechanisms to find such malware. Furthermore it allows for potential improvements to some of the detection techniques.

The conducted experiment showed a detailed methodology of how live memory can be analyzed with the help of the described tools. This methodology can be directly applied on real life cases allowing forensic specialists to extract credentials data from a RAM dump on OS X, potentially obtaining passwords to access otherwise inaccessible parts of data such as C&C (Command & Control) servers, etc.

The described experiment can be expanded in the future to reflect rootkit detection by obtaining a sample rootkit infected RAM dump. This way we could fully explore the available options and performance of the developed plugins. Furthermore we could expand our current analysis by searching additional application related data (calendar events, notes, etc.) that could prove useful in a forensic examination.

6. REFERENCES

[1] Chainbreaker - Mac OS X Keychain Forensic Tool. https://github.com/n0fate/chainbreaker. [Online; accessed May 2016].

[2] The chromium projects. https://www.chromium.org/. [Online; accessed May 2016].

[3] Db browser for sqlite. http://sqlitebrowser.org/. [Online; accessed May 2016].

[4] Developer survey results 2016. http://stackoverflow.com/research/developer-survey-2016technology-desktop-operating-system. [Online; accessed May 2016].

[5] MBR rootkit never dies! news.saferbytes.it/analisi/2012/06/sinowal-mbr-rootkit-never-dies-and-it-alwaysbrings-some-new-clever-features/. [Online; accessed May 2016].

[6] Physical memory forensics. https://www.blackhat.com/presentations/bh-usa-06/BH-US-06-Burdach.pdf. [Online; accessed May 2016].

[7] pycookiecheat - borrow cookies from your browser's authenticated session for use in python scripts. https://github.com/n8henrie/pycookiecheat. [Online; accessed May 2016].

[8] Rekall memory forensic framework. http://www.rekall-forensic.com/pages/at_a_glance.html. [Online; accessed May 2016].

[9] Stat counter global stats - top 7 desktop OSs. http://gs.statcounter.com/os-ww-monthly-201404-201604. [Online; accessed May 2016].

[10] A. Case. Mac memory analysis with Volatility. *DFIR Summit*, 2012.

[11] A. Case. Advancing Mac OS X rootkit detection. 2015.

[12] A. Case, A. Cristina, L. Marziale, G. G. Richard, and V. Roussev. FACE: Automated digital evidence discovery and correlation. *Digital investigation*, 2008.

[13] T. Coppock and D. Gan. Forensic investigation into Mac OS X volatile memory.

[14] D. Farmer and W. Venema. *Forensic discovery*, volume 6. Addison-Wesley Upper Saddle River, 2005.

[15] Kaspersky. Kaspersky lab uncovers "the mask". 2014.

[16] T. Katsuki. Crisis: the advanced malware. 2012.

[17] M. Ligh. Stuxnet's footprint in memory with Volatility 2.0. *MNIN security Blog*, 2011.

[18] L. Myers. New OS X malware: another tibet variant found. 1993.

[19] A. Reyes, K. O'Shea, J. Steele, J. R. Hansen, B. R, Jean, and T. Ralph. *Cyber Crime Investigations*. Elsevier Inc., 2007.

[20] M. Suiche. Advanced Mac OS X physical memory analysis. https://www.blackhat.com/presentations/bh-dc-10/Suiche_Matthieu/Blackhat-DC-2010-Advanced-Mac-OS-X-Physical-Memory-Analysis-slides.pdf. [Online; accessed May 2016].

Pridobivanje vsebine pomnilnika na napravah z operacijskim sistemom Android z uporabo protokolov za posodabljanje strojne programske opreme

Analiza članka pri predmetu Računalniška forenzika*

Klemen Košir
Univerza v Ljubljani
Fakulteta za računalništvo in informatiko
klemen.kosir@kream.io

Nejc Sušin
Univerza v Ljubljani
Fakulteta za računalništvo in informatiko
nejc.susin@gmail.com

POVZETEK

V digitalni forenziki že nekaj časa vse bolj pomembna postaja forenzika mobilnih naprav. Pametni telefoni ne hranijo več le podatkov o klicih, temveč tudi uporabnikovo pošto, slike, lokacijo, koledar, zgodovino brskanja, osebne in finančne podatke ter uporabniška gesla do drugih storitev. Skratka, ogromno podatkov o uporabnikovem početju, ki bi lahko služila kot dokazno gradivo na sodišču.

Skupaj z vse pomembnejšo vlogo mobilnih naprav pa se je izboljševala tudi njihova varnost. Pridobitev podatkov z mobilnih naprav je čedalje težja in potrebno je iskanje novih metod, ki lahko zaobidejo varnostne ukrepe.

Seung Jei Yang in soavtorji z inštituta ETRI (angl. *Electronics and Telecommunications Research Institute*) so v članku z naslovom *New acquisition method based on firmware update protocols for Android smartphones* predstavili novo metodo za pridobitev vsebine pomnilnika mobilne naprave z operacijskim sistemom Android brez izgube oziroma spremembe celovitosti podatkov. Z obratnim inženirstvom (angl. *reverse engineering*) postopka posodabljanja strojne programske opreme so odkrili novo metodo zajemanja, s katero lahko s pomočjo protokolov, ki jih sicer uporabljajo proizvajalci mobilnih naprav, neposredno dostopajo do pomnilnika.

Pristop se izkaže za uspešnega in se izogne večini težav, ki se pojavljajo pri zajemanju podatkov z obstoječimi metodami.

Ključne besede

pridobivanje podatkov, Android, USB, posodobitev, strojna programska oprema

*Študijsko leto 2015/2016. Predmet sta izvajala dr. Andrej Brodnik in dr. Gašper Fele-Žorž.

1. UVOD

Večinski delež mobilnih naprav na trgu je osnovanih na operacijskem sistemu Android. Trg Androidnih naprav pravzaprav že presega trg osebnih računalnikov, rast pa se še ne ustavlja, saj smo vsako leto priča novim tehnologijam, ki dodatno povečajo vlogo mobilnih naprav v vsakdanjem življenju.

V prihodnosti bodo mobilne naprave igrale še večjo vlogo, podatki, ki jih naprave zajamejo bodisi z uporabnikovo vednostjo bodisi brez nje, pa bodo še bolj pomembni za razrešitev sporov in dokazovanje krivde ali nedolžnosti.

Pridobivanje podatkov iz mobilnih naprav tako postane pomembno opravilo, ki pa ima različne ovire. Številna obstoječa orodja se poslužujejo ranljivosti v jedru operacijskega sistema (angl. *kernel*). Seveda se varnostne luknje v sistemu z vsako posodobitvijo popravijo, zato je potrebno iskanje novih, ki so ponavadi vse bolj zapletene. Ranljivosti, ki bi omogočale pridobitev vsebine pomnilnika, trenutno ne poznamo.

Težave pri drugih metodah, na primer s pomočjo obnovitvenih slik (angl. *recovery image*), se pojavijo pri ohranjanju celovitosti podatkov, saj ne moremo zagotoviti, da smo uspešno in brez sprememb pridobili vsebino celotnega pomnilnika. Poleg tega pa obstoječa forenzična orodja za povezavo z napravo uporabljajo orodje *adb* (angl. *Android Debug Bridge*), ki pa ne omogoča pridobitve podatkov, kadar je naprava zaklenjena.

Predlagana metoda pridobitve podatkov zaobide omenjene ovire, saj gre za popolnoma drugačen pristop preko posodobitve strojne programske opreme (angl. *firmware*).

2. PREGLED PODROČJA

Povzeti članek [5] predstavi povsem nov način za pridobivanje podatkov iz pomnilnika mobilnih naprav. Razumljivo je, da je veliko povezav v virih članka povsem razvijalske narave (orodja in predstavitve različnih tehnologij), saj gre za opis dejanskega postopka in ne za znanstveno raziskavo. Vredno je omeniti, da je nemalo teh povezav neveljavnih, kar je nenavadno, glede na to, da je članek izšel le eno leto nazaj.

Breeuwsma je leta 2007 v svojem članku [1] opisal pridobivanje podatkov neposredno iz čipov pomnilnikov *Flash*. Avtor se osredotoči na metode s strojno opremo, saj iz naprav fizično odstrani čipe oziroma uporablja vmesnik *JTAG*. Prav tako velik del nameni analizi nestandardnih datotečnih sistemov. Ker operacijski sistem Android uporablja standardne datotečne sisteme, kot sta *YAFFS2* in družina *EXT*, je analiza modernih slik pomnilnikov veliko lažja.

Leta 2009 sta *Lessard* in *Kessler* [4] pisala o pridobivanju podatkov z mobilnih naprav z eno izmed prvih različic operacijskega sistema Android. V njunem pristopu je bilo potrebno mobilno napravo povezati z računalnikom preko vmesnika USB in ga odkleniti (angl. *rooting*), kar je omogočilo dostop do sistemskih datotek. Prav tako je bilo potrebno onemogočiti zapisovanje podatkov z namensko napravo (angl. *write blocker*). Nato lahko podatke z uporabo programa *FTK Imager* preprosto kopiramo. Analiza podatkov poteka enako kot na navadnih računalnikih.

Kasneje, leta 2013, se je *Glover* v svojem članku [2] osredotočil na popolnoma drugačen pristop. Namesto, da bi podatke pridobil neposredno iz pomnilnika mobilne naprave, je pregledal mobilne aplikacije, s katerimi lahko za uporabniki vohunimo. Pri tem se seveda pojavi več težav, kot je na primer nelegalnost vohunjenja, na nivoju operacijskega sistema Android pa težave s posodobitvami, odklepom telefona in občasno samo strukturo ter arhitekturo sistema. Velika večina podatkov, pridobljenih s to metodo je tudi neuporabnih na sodišču, saj so bile verjetno nelegalno pridobljene, obenem pa ne zagotavljajo nikakršne celovitosti.

3. PRIDOBIVANJE PODATKOV

Mobilne naprave podatke hranijo na različnih medijih, kot so na primer kartice SD, omrežje ali notranje pomnilne enote. Zanimajo nas predvsem slednje, saj se tu nahaja večina podatkov o uporabnikovi dejavnosti. Posebej zanimivi so podatki v kratkoročnem pomnilniku, saj tu lahko najdemo uporabniška imena, gesla, ključe za šifriranje in podatke o aplikacijah, ki so nameščene na napravi.

Pri pridobivanju vsebine pomnilnika RAM je potrebna posebna previdnost, saj ne želimo, da bi se podatki spremenili. Priporočen način zajemanja podatkov je zato med samim delovanjem naprave (*live acquisition*), saj tako dobimo največ podatkov, poleg tega pa se izognemo morebitnemu šifriranju podatkov.

3.1 Obstoječe metode zajema podatkov

Za pridobitev podatkov iz mobilnih naprav se uporabljata dve glavni metodi — pridobitev s programsko opremo (angl. *software-based*) ali pa s strojno opremo (angl. *hardware-based*).

Pridobivanje s programsko opremo lahko dodatno razdelimo na dve vrsti [3]. Logična pridobitev (angl. *logical acquisition*) pridobi podatke iz strukture datotečnega sistema, kar pomeni, da iz razdelkov v pomnilniku (na primer `BOOT`, `RECOVERY`, `SYSTEM`, `CACHE` in `USERDATA`) prenesemo obstoječe datoteke in mape. Tako lahko pridobimo le uporabnikove podatke, na primer slike in zgodovino klicev, ne pa izbrisanih datotek.

Fizična pridobitev (angl. *physical acquisition*) prenese podatke neposredno iz pomnilnika mobilne naprave preko priključka USB. Prenešeni podatki so popolna kopija pomnilne naprave. Za izvedbo fizične pridobitve potrebujemo administrativni dostop, zato moramo napravo najprej odkleniti (angl. *rooting*), poleg tega pa omogočiti razhroščevalni način (angl. *USB debugging*) [4]. Odklepanje mobilne naprave poteka tako, da na datotečni sistem dodamo program `/system/su`, s katerim lahko drugim programom omogočimo dostop do sistemskih datotek. Zaradi tega se te metode srečajo s težavami s posodobitvami, ki lahko popravijo varnostne luknje, ki jih uporabljajo metode za odklep mobilnih naprav. Prav tako se tukaj ne ohranja celovitost podatkov.

Pridobivanje s strojno opremo, kjer podatke pridobimo neposredno iz čipa pomnilne naprave, ima dva glavna predstavnika — vmesnik *JTAG* [1] in metoda *chip-off*. Vmesnik JTAG (angl. *Joint Test Action Group*) se je izkazal kot zelo uspešnega, vendar le-ta ni prisoten na vseh mobilnih napravah, poleg tega pa je pridobivanje časovno potratno, saj lahko traja tudi do osem ur. Ta vmesnik lahko uporabimo s programerji, posebno strojno opremo, ki omogoča prenašanje in zapisovanje podatkov v obstojni pomnilnik (angl. *non-volatile random-access memory*. Poleg tega lahko vmesnik uporabljamo za razhroščevanje kot tudi nadzor temperature mobilne naprave. Vmesnik se zadnje čase uporablja tudi pri izdelavi vezij FPGA (angl. *field-programmable gate array*).

Pri drugi metodi pridobivanja s strojno opremo gre za fizično odstranitev čipa in branje surovih podatkov, kar je uporabno le v redkih primerih, ko smo napravo zmožni fizično razstaviti.

3.2 Posodobitev strojne programske opreme

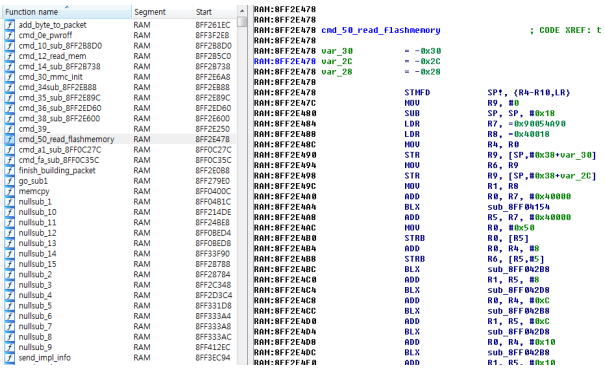
Za posodobitev strojne programske opreme mobilnih naprav moramo uporabiti programe proizvajalcev naprav (na primer LG, Sony, HTC, Samsung ...), saj le-ti uporabljajo protokole, katerih specifikacije niso javno znane.

Med samo posodobitvijo se mobilna naprava zažene v poseben način, imenovan *firmware update mode* ali *download mode*. V tem načinu so dejavni le notranji pomnilnik, zaganjalnik (angl. *bootloader*) in krmilnik USB. Ta način delovanja lahko s pravilnim zaporedjem pritiskov gumbov zaženemo kadarkoli. V primeru, da mobilne naprave na moramo zagnati v tem načinu s pomočjo pritiskov gumbov, lahko uporabimo ustrezen kabel *USB Jig*.

V povzetem članku [5] so avtorji z analizo delovanja programov odkrili ukaze, s katerimi lahko v načinu za posodabljanje dostopajo do pomnilnika, saj morajo le-ti omogočiti branje in pisanje, da lahko na napravo naložijo novo različico strojne programske opreme. Njihov cilj je bil, da ugotovijo, kaj omogočajo posamezni ukazi in kako lahko z njimi pridobijo vsebino pomnilnika mobilnih naprav.

Opisana je bila analiza mobilnih naprav proizvajalcev LG, Pantech in Samsung, vendar je avtorjem na koncu uspelo uspešno pridobiti podatke iz 80 modelov mobilnih naprav različnih proizvajalcev. Pri tem moramo upoštevati, da zaganjanje mobilnih naprav v način za posodabljanje preko pritiskov gumbov na Androidu deluje le do različice 5.0.

Do samih ukazov in njihovega delovanja na mobilnih napravah proizvajalca LG so avtorji prišli z razgradnjo (angl. *disassembly*) in obratnim inženirstvom nalagalnika *SBL3* z uporabo programa *IDA Pro*. Primer zbirne kode ukaza `0x50` prikazuje slika 1.



Slika 1: Analiza ukaza 0x50 v programu IDA Pro.

3.3 Pridobivanje podatkov z naprav LG

Avtorji članka so za pridobivanje podatkov iz telefonov proizvajalca *LG* analizirali program *LG PC Suite*.

Ukazi, ki jih pošilja navedeni program, so v obliki paketa HDLC (angl. *High-Level Data Link Control)*, kar je prikazano v tabeli 1. Prvi in zadnji bajt paketa sta zastavici `7E`, s katerima označimo začetek in konec ukaza. V paketu prav tako dodamo bajt, s katerim povemo, kateri ukaz želimo izvesti, in dodatni podatki, ki so dolgi poljubno število bajtov. Na koncu sledi še nadzorna vsota *CRC-16 X.25-CCITT*, s katero lahko mobilna naprava preveri, če je bil paket ustrezno prenešen.

Tabela 1: Oblika ukazov za mobilne naprave LG.

0x7E	ukaz	podatki	CRC	0x7E
8 bitov	8 bitov	n * 8 bitov	16 bitov	8 bitov

Postopek in ukazi za nadgradnjo strojne programske opreme mobilnih naprav proizvajalca LG so sledeči:

1. `0x00`: program pridobi podatke o napravi,
2. `0x3A`: naprava se zažene v načinu za posodabljanje,
3. `0x2F`: pridobimo podatke o različici protokolov, trenutni programski opremi,
4. `0x30`: preberemo strukturo razdelkov na disku,
5. `0xFA`: program prebere tovarniške podatke,
6. `0x39`: v pomnilnik zapišemo sektor (ta korak se ponavlja, dokler se strojna programska oprema v celoti ne prepiše z novo različico),
7. `0x38`: mobilni napravi sporočimo, da je posodobitev končana,
8. `0x0A`: ponovno zaženemo mobilno napravo.

Poleg same analize programa za posodabljanje je bilo potrebno analizirati tudi delovanje zaganjalnika (angl. *bootloader*) mobilne naprave, iz katere želimo pridobiti vsebino pomnilnika. Koda nalagalnika je ponavadi priložena paketu s strojno programsko opremo. V kodi so avtorji našli natančno obliko ukaza, ki se pošlje znotraj paketa HDLC, kar prikazuje tabela 2. Ukazu moramo podati začetni naslov, s katerega želimo brati in velikost prebranih podatkov. Naslov lahko pridobimo iz mobilne naprave z ukazom `0x30`, s katerim preberemo podatke o pomnilniku in posameznih razdelkih.

Tabela 2: Oblika ukazov za branje podatkov.

0x50	ukaz	naslov	velikost	CRC
8 bitov		32 bitov	32 bitov	16 bitov

Spodnji izsek prikazuje zahtevek in odgovor primera ukaza, s katerim lahko preberemo preberemo nekaj podatkov iz mobilne naprave:

```
Zahtevek:
7E 50 01 01 00 00 00 00 00 00 80 02 00 00
02 00 00 00 00 00 00 00 00 00 00 7C 1B 7E

Odgovor:
50 01 01 00 00 00 00 00 00 80 02 00 00 02
00 00 04 E7 03 00 00 00 00 00 E0 1C 41 4E
44 52 4F 49 44 21 A0 00 74 00 00 80 20 80
63 93 29 00 00 00 20 82 00 00 00 00 00 00
10 81 00 01 20 80 00 08 00 00 00 00 00 00
38 06 FA E0 89 76 6D 61 6C 6C
0M console=ttyHSL0,115200,n8 lpj=67677 us
er_debug=31 msm_rtb.filter=0x0 ehci-hcd.p
ark=3 coresight-etm.boot_enable=0 android
boot.hardware=geehrc8
```

Avtorji so po več zaporednih zajemih podatkov z razpršilno funkcijo preverili še celovitost podatkov. Ugotovili so, da s to metodo ni prišlo do nikakršne spremembe. S predlagano metodo zajema lahko v 30 minutah prenesemo 32 GB podatkov, kar je približno štirikrat hitrejše od drugih metod. Poleg same analize so avtorji članka napisali tudi program, imenovan *Android Physical Dump*, ki omogoča pridobivanje podatkov iz vseh mobilnih naprav, ki so združljive s protokoli, za katere so našli ustrezne ukaze.

Pošiljanje ukazov na mobilne naprave drugih proizvajalcev je podobno, vendar se razlikuje po obliki paketov oziroma ukazov, ki jih pošljemo napravi preko krmilnika USB.

3.4 Pridobivanje podatkov z naprav Pantech

Poleg mobilnih naprav LG so se avtorji posvetili tudi proizvajalcu Pantech. Tako kot pri mobilnih napravah LG, se tudi pri mobilnih napravah Pantech ukazi skrivajo v nalagalniku. Tokrat so avtorji analizirali nalagalnik *ABOOT*.

Postopek za posodobitev strojne programske opreme mobilnih naprav Pantech je sledeč:

1. `AT*PHONEINFO`: program pridobi podatke o napravi,
2. `AT*PDL*START`: naprava se zažene v načinu za posodabljanje,

3. `0x00`: začetek postopka posodobitve,
4. `0x02`: naprava se pripravi na zapisovanje,
5. `0x04`: program pobriše podatke na podanem naslovu,
6. `0x05`: program zapiše podatke na podani naslov,
7. `0x03`: konec postopka posodobitve,
8. `0x01`: ponovno zaženemo napravo.

Program za posodabljanje napravi pošlje ukaz velikosti 32 bajtov, katerega oblika je prikazana v tabeli 3. Novejše naprave uporabljajo pakete velikosti 128 bajtov.

Tabela 3: Oblika ukaza za branje.

0x06	razdelek	0x00	naslov	velikost	0x00
32 bitov	32 bitov	32 bitov	32 bitov	32 bitov	96 bitov

Ukaz, ki ga lahko uporabimo za branje z naprave je `0x06`. Poleg samega ukaza potrebujemo tudi številko razdelka, do katerega dostopamo, začetni naslov in velikost prebranih podatkov. Preden lahko izvedemo postopek pridobitve podatkov, moramo ugotoviti številko razdelka. Mobilne naprave Pantech nimajo podobnega ukaza kot mobilne naprave LG, zato moramo razdelke poiskati sami. To storimo tako, da z ukazom `0x06` preberemo vsebino razdelka GPT (angl. *GUID Partition Table*) s številko `0x0A`, ki hrani podatke o vseh ostalih razdelkih.

3.5 Pridobivanje podatkov z naprav Samsung

Tretji proizvajalec, katerih mobilne naprave so avtorji članka obravnavali, je Samsung. Postopek analize je bil enak, tako da so avtorji našli ustrezen ukaz za branje, vendar so odkrili, da je bila iz nalagalnika odstranjena koda za branje iz pomnilnika.

Kljub temu so analizirali postopek za posodobitev strojne programske opreme, v katerem so pošiljajo paketi velikosti 1024 bajtov z uporabo 32-bitnih ukazov.

Postopek za posodobitev strojne programske opreme mobilnih naprav Samsung je sledeč:

1. `0x64`: program inicializira protokol in začne postopek posodobitve,
2. `0x65`: na mobilno napravo se prenese tabela razdelkov posodobljene strojne programske opreme,
3. `0x66`: posodobitev se zapiše v pomnilnik mobilne naprave,
4. `0x67`: program zaključi postopek posodobitve.

Ukaz `0x66` vsebuje tudi podukaza `0x01` in `0x03`, s katerimi bi lahko pridobili podatke iz pomnilnika, vendar se naprava le odzove z `ACK`.

Ker je bila strojna koda iz nalagalnika odstranjena, avtorjem ni uspelo pridobiti podatkov z mobilnih naprav Samsung.

4. ZAKLJUČEK

Predstavljena metoda pridobivanja vsebine pomnilnika iz mobilnih naprav z operacijskim sistemom Android s pomočjo protokola za posodobitev strojne programske opreme se izkaže za zelo uspešno. Postopek ni preprost na vseh napravah. Protokol določajo proizvajalci, zato je potrebno veliko prilagajanja.

Preden lahko izvedemo sam postopek, moramo analizirati programe za posodobitve mobilnih naprav posameznega proizvajalca in ustrezno določiti delovanje ukazov. Nekateri proizvajalci iz posodobitvenih programov odstranijo ukaze za branje pomnilnika, zato je v teh primerih potrebno dodatno delo. Avtorji povzetega članka so uspešno pridobili vsebino pomnilnika iz več kot 80 modelov naprav različnih proizvajalcev.

Težave, predstavljene v uvodu, ne vplivajo na predlagano metodo. Napravo lahko v vsakem trenutku zaženemo v načinu za posodobitev, brez potrebe po razvijalskih ali administratorskih pravicah. Pridobimo celotno vsebino pomnilnika, ki se med izvajanjem ne spreminjado saj zagotavlja celovitost podatkov. Edina omejitev je, da je potrebno za vsako novo različico operacijskega sistema Android in model telefona analizirati protokol in proces posodobitve strojne programske opreme ter preveriti, če je metoda sploh izvedljiva.

5. VIRI IN LITERATURA

[1] M. Breeuwsma, M. De Jongh, C. Klaver, R. Van Der Knijff, and M. Roeloffs. Forensic data recovery from flash memory. *Small Scale Digital Device Forensics Journal*, 1(1):1–17, 2007.

[2] J. Grover. Android forensics: Automated data collection and reporting from a mobile device. *Digital Investigation*, 10:S12–S20, 2013.

[3] Z. Jovanovic and I. Redd. Android forensics techniques. *International Academy of Design and Technology*, 2012.

[4] J. Lessard and G. Kessler. Android forensics: Simplifying cell phone examinations. 2010.

[5] S. J. Yang, J. H. Choi, K. B. Kim, and T. Chang. New acquisition method based on firmware update protocols for android smartphones. *Digital Investigation*, 14:S68–S76, 2015.

Reliable and Trustworthy Memory Acquisition on Smartphones

Tjaž Brelih
Faculty of Computer and Information Science, University of Ljubljana
tb3383@student.uni-lj.si

Matej Ulčar
Faculty of Computer and Information Science, University of Ljubljana
mu5618@student.uni-lj.si

Jani Bevk
Faculty of Computer and Information Science, University of Ljubljana
jb1475@student.uni-lj.si

ABSTRACT

More and more malicious code is emerging every day and an insignificant chunk of it is targeting smartphones. The malware and its effects on a system must first be analysed in order to combat it effectively. In this paper we review TrustDump, a mechanism for reliably obtaining the contents of the RAM and CPU registers of a possibly compromised operating system. It is based on ARM's TrustZone technology, where the OS being analysed is running in the normal world and the analysis tools are running in the secure world. The latter has higher privileges and is separated from the former on the hardware level. In this work we describe the TrustDump mechanism and overview some possible alternatives.

Keywords

TrustZone, TrustDump, smartphone memory acquisition, Trusted Execution Environment, malware analysis

1. INTRODUCTION

Smartphones have nowadays become a commodity and are commonly used for both personal and business purposes. This makes them an attractive target for conducting privacy and security attacks. Malicious code might not be as widespread on smartphones as it is on personal computers but the threat exists. Malware analysis is a crucial step in developing anti-malware tools.

There are two approaches to malware analysis: the *in-the-box* approach and the *out-of-the-box* approach [7]. For the in-the-box approach the malware analysis tools are installed in the same OS as the malware. This makes it very efficient at studying the malware's behaviour, however, it is vulnerable to advanced malware that might detect it and tamper with the analysis. For the out-of-the-box approach the malware analysis tools are installed into an environment which is securely separated from the OS containing the malware. Examples of these isolated environments are Virtual Machines and Trusted Execution Environments.

On ARM processors, which are widely used in smartphones, a Trusted Execution Environment can be established using the TrustZone technology, described in section 3.1. A mechanism, called TrustDump, has been developed [7]. It relies heavily on the TrustZone technology and enables the memory acquisition of the OS being analysed. Memory analysis is an important step in understanding the inner workings of a specific malware. We describe this mechanism in section 3.2.

2. RELATED WORK

2.1 Software-based Solutions

Software-based solutions provide a simple to use and efficient way of memory acquisition. Their main downfall, however, is that they rely on host OS (usually Android OS) or a hypervisor to acquire the OS memory. In the case that the host OS or a hypervisor have been compromised, a reliable memory acquisition can not be guaranteed.

One such solution for Android OS is Linux Memory Extraction (LiME), which was developed due to `/dev/mem` device being recently disabled because of security concerns. LiME uses a loadable kernel module, which is transferred to the Android device and executed. The problem with this is that the Linux kernel uses a security mechanism, which prevents inserting incompatible code into the operating system. Therefore a generic approach is not possible, but a customized module has to be built for each smartphone and Android version. Another obstacle using LiME, or indeed any software-based solution, is that the examiner should first gain root privileges. The technique to gain root privileges, again, can vary for every device and Android version and Linux kernel in use [6].

Another software-based solution is to use a hypervisor (also called virtual machine monitor) to bypass the host OS. By doing so, one can prevent the host OS from interfering with the memory acquisition, should it be compromised. Using Linux's Kernel-based Virtual Machine (KVM) as a basis, KVM/ARM has been developed for virtualization on ARM processors. It can be used for virtual machine inspection technique on smartphones. By integrating KVM/ARM with Linux, it is available on any device running a recent version of the Linux kernel. Hypervisors running on bare metal (eg. Xen) must actively support every platform on which they are to be run [5].

2.2 Hardware-based Solutions

An important benefit of hardware-based solutions is that they access the smartphone's hardware directly, without utilizing the host OS in any way. That way, they are completely protected from a potentially malicious code in the host OS. They also do not write to memory or disk, so they cannot accidentally alter any evidence regarding the attack or malware.

Brian D. Carrier and Joe Grand designed a procedure using a PCI expansion card. PCI is independent from the host OS or any software, so while the attacker may see it, they cannot tamper with it or intercept any of its actions. A physical switch on the PCI card activates the card which first suspends the CPU, so that the memory state does not change, and then transfers the data stored in memory to an external device. CPU is then resumed after the transfer has been completed. A major obstacle with this procedure is that the PCI card needs to be installed prior to an incident taking place. Once installed, however, the memory acquisition is simple and fast (simply pressing a button) and can be done by the victim during the attack, not involving a third party [4].

Two rather destructive methods of accessing the memory chips directly are JTAG (Joint Test Action Group) and Chip-off. JTAG normally uses test access ports of the printed circuit board for testing or debugging embedded software. JTAG test access port can also be used to access memory directly, but it requires a lot of knowledge and soldering skills. No mistakes are permitted otherwise the data can be lost [3, 8]. Chip-off technique consists of physically removing the NAND flash chip. The removal damages the balls on the chip so they need to be repaired first. A special equipment is then needed to read the chip, which can be very costly [8].

3. TRUSTDUMP

3.1 TrustZone

TrustZone [2] is a set of security extensions for the ARM platform. It is supported by a broad range of processors from ARMv6 architecture onward, including the ARM11 and Cortex-A families, most often found in devices with embedded systems.

It provides a hardware level isolation between two execution worlds: *secure (trusted) world* and *normal (non-trusted) world* [1]. The secure world has a higher access privilege than the normal world and can therefore access the resources of the normal world and inspect the software running on it. The reverse is, however, not true. The normal world is restricted and cannot access anything within the secure world. This prevents information leaking from the trusted world to the less trusted world. The split between the two worlds, shown in figure 1, is orthogonal to other processor modes and privilege levels, so both worlds can operate independently of each other on the same core.

The current state of the CPU is indicated by the NS (*non-secure*) bit in the Secure Configuration Register of the CPU. A zero corresponds to the secure world and a one to the normal world. The switch between the two worlds occurs with the help of an additional CPU mode, called *monitor mode*. In this mode the CPU is always executing in the secure world regardless of the NS bit value. The instructions that execute within monitor mode generally (depends on implementation) save the state of the current world and restore the state of the other world. This allows a single physical processor core to safely execute code from both worlds and thus removes the need for a dedicated security processor core. The world switch is only possible through the monitor mode and can be triggered by executing a special *Secure Monitor Call* instruction or by interrupts and other hardware exception mechanisms. This split between secure and normal world is not limited to just the CPU. The system bus contains additional NS bits which propagate the current state through transactions on the bus to the memory and peripheral devices [1].

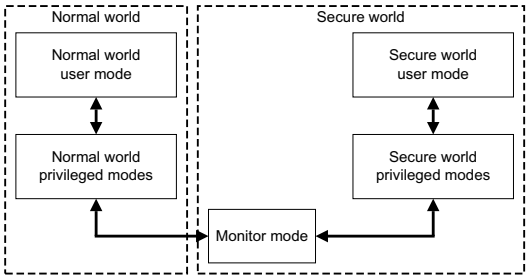


Figure 1: TrustZone execution worlds [1].

TrustZone is commonly used to provide a certifiable Trusted Execution Environment (TEE) which is completely separated from the main operating system (also called a *Rich OS*) running in the normal world. The device's boot process must of course also be secure, which is usually done by cryptographically verifying each step of the boot process [1]. Typical applications of the TrustZone technology are cryptography, Digital Rights Management, and protection of authentication mechanisms and keys. For example, the Rich OS can ask the secure world to encrypt some data on its behalf using keys that are only accessible from software running in the secure world. The keys are safely protected from any malware that might be running in the normal world.

There are multiple possible software architectures for the secure world, the most popular for Android being the TEE implementation by Trustonic and Qualcomm's QSEE. ARM also has its own reference implementation. In some implementations certified trusted applications can be installed to the secure world in order to allow for greater extensibility and a wide variety of authentication options. Similar technologies to TEE exist for other processor architectures, e.g. Intel's TXT and AMD's Secure Execution Environment.

3.2 TrustDumper

TrustDumper is a memory acquisition software module, part of the TrustDump framework, that is installed in the secure domain which performs data acquisition and analysis of the Rich OS's memory and CPU registers. On the ARM platform the operating system runs in the secure domain, but, for the TrustDump mechanism, it needs to be ported to the normal domain, to allow TrustDumper to run in the secure domain. TrustDumper runs in a sealed memory region,

which OS, running in the normal domain, can not access. The OS running in the normal domain is frozen when the system enters the secure domain.

TrustDump is OS agnostic (it supports memory acquisition of any operating system), so no changes need to be made to the OS that the data is extracted from. This is important for forensics, because it ensures that data does not alter before the extraction. Furthermore, TrustDumper is self-contained, meaning there is no need to install an operating system in the secure domain.

TrustDump supports immediate malware detection, as well as exporting data for later malware analysis. TrustDumper runs independent of the Rich OS, so it needs to fill the semantic gaps to successfully detect the malware and analyse it in the secure domain. It accomplishes this by reconstructing the Rich OS context knowing the kernel data structures. For offline analysis of acquired memory, the data is transferred via serial port or a network card along with its hash value to check if the data has been successfully transferred.

To ensure secure and reliable memory acquisition, switching into secure domain must be reliable. It should always happen upon user's request, even if the Rich OS has been compromised or if it simply crashes. Thus, one can never depend on Rich OS for reliable switching. To satisfy these demands, TrustZone provides two ways of switching into secure domain from the normal domain: SMC instruction and Secure Interrupt. SMC instruction can only be run from Rich OS's kernel, so if the Rich OS is compromised, it can intercept and/or block these instructions. TrustZone's secure interrupts can be called instead to reliably switch from the normal to the secure domain. Mobile platforms are widely using non-maskable interrupts, triggered by a certain button or a combination of buttons. Since these non-maskable interrupts can not be intercepted by the Rich OS, they can be used for safe system switching. Additionally, a secure interrupt can be configured as a non-maskable interrupt on platforms without one.

When the non-maskable interrupt is triggered by a physical button, TrustDump freezes the Rich OS and safely switches from normal to secure domain, regardless of the state of the Rich OS. By doing so, malware can not hide its traces and can thus be detected, while at the same time it also can not interfere with the system switch or memory acquisition, because the Rich OS is frozen and because TrustZone completely isolates TrustDumper in secure domain from the Rich OS.

3.3 TrustDump Implementation

The authors of the article used a modified version of Android 2.3.4 as the Rich OS [7]. The modification allowed Android to run only in the normal domain, since it utilizes secure domain to gain access to some of the peripherals, namely the Deep Sleep Mode Interrupt Holdoff Register. This register is used to hold off the interrupts before the CPU enters the low power mode. The authors implemented functions *secure_write* and *secure_read* which allow the modified Android to write or read to secure peripherals from the normal domain. *secure_write* writes a 32-bit *data* to a physical address *pa*, while *secure_read* reads from the physical address *pa* and returns the 32-bit data found at that address.

Since we can not rely on the Rich OS to switch into secure domain, the TrustDump framework has to configure and use special interrupts to do so. TrustZone enabled chips use a TrustZone Aware Interrupt Controller (TZIC) which receives interrupts from peripheral devices and routes them to the ARM processor. By default, the TZIC uses Fast Interrupt (FIQ) as a secure interrupt and Regular Interrupt (IRQ) as a non-secure interrupt. There are four steps necessary for a reliable switch from the normal domain to the secure domain to be made. First, a peripheral device makes an interrupt request. Second, this interrupt request is sent to the TZIC. Third, TZIC sends a corresponding exception to the CPU based on the type of the incoming interrupt (FIQ or IRQ). Finally, the CPU makes the switch from normal domain to the secure domain.

All of the above steps are critical for a reliable switch from normal to secure domain. A compromise at any stage of the switch can result in the switch not occurring. If the source of the interrupt can be masked by the Rich OS or the Rich OS just blocks all FIQ requests, then the switch to the secure domain will be blocked. To prevent this from happening the authors constructed a non-maskable interrupt using a button connected through a GPIO peripheral device. This GPIO device must be set as secure so it can send FIQ requests to the TZIC which makes it impossible for the normal domain to access the GPIO device. To circumvent this problem the authors used the aforementioned *secure_write* and *secure_read* functions. The list of peripherals that can be accessed this way is contained in a *Whitelist*. This way the authors achieved the protection of the source of non-maskable interrupt from the normal domain without constricting the access of the normal domain to other devices located on the same GPIO device.

After we make the switch to the secure domain TrustDumper starts the process of dumping normal domain registers and memory. ARM processors use banked registers, which means that normal and secure domain each have their own copy of registers. While normal domain can only access its own registers, secure domain can access both its own and registers from the normal domain. We can also access the entirety of physical Rich OS's physical memory from the secure domain. However, to access Rich OS's virtual addresses the TrustDumper must use the page tables of the Rich OS to get the corresponding physical addresses. Memory dumping involves transmitting data from RAM to peripherals. To speed up the data transmission TrustDumper uses DMA (*direct memory access*) controller. DMA allows us to transfer data from RAM to a peripheral device without the involvement of CPU. To ensure that the state of the system does not change during the memory dump phase TestDumper saves the current state of the DMA before reconfiguration. After TestDumper finishes with the memory dump the previous state of DMA is restored. In their prototype the authors used serial port as the peripheral to transmit RAM to external computer.

4. CONCLUSIONS

Various methods of memory acquisition on smartphones have been developed throughout the years. While software-based methods seem simple and straight-forward, they do not offer a completely reliable and trustworthy memory acquisition, because they can be tampered with. On the other hand, hardware-based methods tend to depend on expensive equipment or complicated preparations. With TrustZone being widely supported by ARM processors, ensuring a hardware level isolation between normal domain and secure domain is now easier. With addition of TrustDump to reliably transfer the memory data, using non-maskable interrupts to minimize the impacts on the Rich OS, memory acquisition is fast, reliable and trustworthy.

5. REFERENCES

[1] ARM security technology building a secure system using TrustZone® technology. `http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.prd29-genc-009492c/index.html`. Accessed: 2016-05-10.

[2] ARM TrustZone. `http://www.arm.com/products/processors/technologies/trustzone/`. Accessed: 2016-05-10.

[3] I. M. Breeuwsma. Forensic imaging of embedded systems using jtag (boundary-scan). *Digital Investigation*, 3(1), 2006.

[4] J. G. Brian D. Carrier. A hardware-based memory acquisition procedure for digital investigations. *Digital Investigation*, 1(1), 2004.

[5] J. N. Christoffer Dall. Kvm/arm: The design and implementation of the linux arm hypervisor. In *Proceedings of the 12th Annual Linux Symposium*, 2010.

[6] A. P. Heriyanto. Procedures and tools for acquisition and analysis of volatile memory on android smartphones. In *Proceedings of the 11th Australian Digital Forensics Conference*, 2013.

[7] H. Sun, K. Sun, Y. Wang, J. Jing, and S. Jajodia. Trustdump: Reliable memory acquisition on smartphones. In *Computer Security-ESORICS 2014*, pages 202–218. Springer, 2014.

[8] I. D. D. R. Zlatko Jovanovic. Android forensics techniques. *International Academy of Design and Technology*, 2012.

Del II

Omrežna forenzika in forenzika mobilnih naprav

Senzorji na pametnih telefonih kot digitalni dokaz

David Možina
Fakulteta za računalništvo in informatiko
Večna pot 113
Ljubljana, Slovenija
dm1711@student.uni-lj.si

Gregor Porocnik
Fakulteta za računalništvo in informatiko
Večna pot 113
Ljubljana, Slovenija
gp3964@student.uni-lj.si

Romana Grilj
Fakulteta za računalništvo in informatiko
Večna pot 113
Ljubljana, Slovenija
romana.grilj@gmail.com

Povzetek

Senzorji v pametnih telefonih omogočajo nove priložnosti za iskanje dokazov v digitalni forenziki. V pametne telefone se vgrajuje vedno več različnih senzorjev, ki beležijo različne podatke o stanju telefona. Podatki, ki jih beleži pametni telefon so v digitalni forenziki lahko uporabni npr. pri potrjevanju ali zavračanju alibija ali pri dokazovanju dejavnosti oz. aktivnosti osumljenca. Za pregled podatkov na pametnih telefonih digitalni forenziki uporabljajo za ta namen izdelano programsko opremo. V tem članku bomo predstavili idejo o zajemu podatkov z različnimi senzorji, kako se te podatke ustrezno pregleda, ter s kakšnimi izzivi se lahko srečamo pri sami pridobitvi podatkov. Na praktičnem primeru bomo prikazali kje in kako na operacijskem sistemu Android najdemo podatke o lokaciji, ter kako jih uporabimo. Kot zanimivost bomo pogledali tudi kako podjetje Google beleži podatke o lokaciji, kdo se jih lahko poslužuje in kako pridemo do le-teh.

Ključne besede

senzorji, dokazi na mobilnih napravah, postopki digitalne preiskave, digitalni dokazi, senzorji na pametnih telefonih

1. UVOD

Širjenje pametnih telefonov in njihova popularnost predstavlja velik izziv digitalnim forenzikom pri odkrivanju kaznivih dejanj in incidentov. Razlog leži predvsem v različnih tipih strojne in programske opreme, ki je proizvedena s strani proizvajalcev pametnih telefonov, ta pa se s časom hitro spreminja. Lahko se zgodi, da so razni dodatki za pametni telefon na voljo le omejen čas, na primer za čas, ko je pametni telefon v redni prodaji. To predstavlja različne ovire za forenzične preiskave, saj je za učinkovito preiskavo potrebno imeti ustrezno opremo, npr. specifikacije ali kable za določen model telefona, težave pa se lahko pojavijo tudi pri pregledu starih naprav, ki niso več podprte s strani proizvajalcev. Podatki, ki jih dobimo s pomočjo senzorjev so lahko koristni pri preiskovanju posameznih kaznivih dejanj v smislu, da zagotavljajo bistvene informacije o samem kaznivem dejanju, uporabni pa so tudi pri odkrivanju in preprečevanju kriminala.

Tovrstno zbiranje podatkov lahko dvigne etična in pravna vprašanja, ki se nanašajo na tajnost telekomunikacijskih operaterjev in zasebnost udeleženih oseb. Zaradi posega v zasebnost in temeljne pravice je taka oblika nadzora in zbiranje teh podatkov mogoča le v primeru, da obstaja sum za visoko stopnjo ogroženosti. Trenutno se zdi, da je digitalna forenzika na mobilni napravah še v začetnih fazah, kljub temu pa obstajajo določena orodja s katerimi lahko zbiramo podatke na različnih platformah oz. operacijskih sistemih.

V nadaljevanju bomo pod drugo točko predstavili samo izvajanje digitalne forenzične preiskave in njene procesne modele, ki se jih glede na tip preiskave uporabi v mobilni forenziki, v tretji točki bomo podrobno opisali senzorje in različne možnosti za zajem podatkov različnih senzorjev, v četrti točki bomo na zaseženem Android telefonu prikazali kako pridemo do podatkov o lokaciji in kako jih uporabimo.

2. IZVAJANJE DIGITALNE FORENZIČNE PREISKAVE

Digitalno forenzično preiskavo lahko sproži kdorkoli. Lahko je to nekdo iz notranjih virov, kot je npr. korporacijska preiskava ali iz zunanjih virov (dokazi bodo podprli oz. ovrgli hipoteze na sodišču). Ko se to zgodi je potebno forenzično preiskavo pravilno izvesti. Digitalna forenzična presikava se izvaja po točno določenih korakih oz. modelih. Procesni modeli za opravljanje digitalnih forenzičnih preiskav opisujejo postopke ki potekajo pred, med in po preiskavi fizičnega ali digitalnega zločina. Te procesi se nanasajo na metodologije in tehnike katere so povezane z zbiranjem, analizo in interpretiranjem digitalnih dokazov in so primerni za forenzično preiskavo. Dokumentacija pri digitalni presikavi je stalen proces in poteka v vseh fazah preiskave. Med samo preiskavo je vsak posameznik, ki je udeležen v preiskavi odgovoren za vse aktivnosti katere so povezane z elektronskimi dokazi. Te morajo biti tudi primerno zaščiteni. Vsak, ki pride v stik z zaseženim gradivom oziroma z zaseženimi podatki, mora biti dokumentiran, pooblaščen in usposobljen za delo z napravami oziroma s podatki, nepooblaščenim osebam pa mora biti dostop do teh podatkov onemogočen.

2.1 Procesni modeli za izvajanje forenzične preiskave

Obstaja več procesnih modelov po katerih lahko izvajamo forenzično preiskavo. Večina modelov se loči po tem, da vsebuje različne aktivnosti ali pa ima različen potek same preiskave.

Na splošno se forenzična preiskava začne izvajati po prejemu obvestila o incidentu. Sestavi se ekipa, ki preveri sam incident, ter se odloči za morebitno kopijo podatkov. Pri izdelavi kopije podatkov je potrebno paziti, da sami podatki ostanejo prav taki kot so in da se med kopiranjem ne spremenijo. Pomembno je, da so digitalne kopije podatkov čimbolj natančne, saj je to pogoj za kasnejšo analizo v laboratoriju. Po ustvarjenih kopijah se lahko priskovalci lotijo raziskovanja samega kraja zločina in tako opazijo še kakšne morebitne dokaze. Ko imajo predstavo o poteku digitalnega zločina, katero so si ustvarili na podlagi zbranih podatkov se prične faza rekonstrukcije incidenta, kjer se pridobljeni dokazi preverijo in oblikujejo v končno zgodbo. Zadnja faza preiskave je predstavitev teorije o tem kako je zločin potekal.

Reith je leta 2002 predlagal model za opravljanje digitalnih forenzičnih preiskav, ki vključuje devet faz: identifikacijo, pripravo, strategija pristopa, ohranjanje, zbiranje podatkov, pregled podatkov, predstavitev podatkov, analiza podatkov ter vračanje dokazov[2].

Casey je predlagal model kjer se na začetku preiskave vključi filtre. S pomočjo filtrov odstranimo podatke, ki pri sami preiskavi nimajo dodane vrednosti (npr. datoteke katere ustvari sistem). Model preiskave je predlagal v sedmih korakih: avtorizacija in priprava, identifikacija, dokumentacija, zbiranje in hramba podatkov, pregled in analiza, rekonstrukcija ter poročanje[3].

Carrier in Spafford sta predlagala model, ki bi temeljil na procesu preiskave samega kraja zločina. Model temelji na tem, da si preiskovalec najprej ogleda kraj zločina. Ogled samega kraja zločina pripore k tem, da preiskovalec dobi celotno sliko in tako lažje oblikuje hipoteze. Model vsebuje naslednjih pet faz: priprava na preiskavo, uvajanje oz. izvajanje preiskave, fizični pregled kraja zločina, digitalni pregled kraja zločina in predstavitev dokazov[4].

2.2 Izvajanje forenzične preiskave na mobilnih napravah

Forenzična preiskava elektronskih mobilnih naprav postaja vse pogostejša in vse pomembnejša. Je področje katerega razvoj je izredno hiter. Pokriva tako področje notranjih preiskav in incidentov v podjetjih, kot tudi preiskave elektronskih naprav za potrebe pridobivanja elektronskih dokazov v pravnih in kazenskih postopkih. Cilj katerekoli forenzične preiskave je poiskati dejstva in z njihovo pomočjo priti do celotne slike dogodka oziroma dogajanja. V pametnih mobilnih napravah se preiskava osredotoča na tehnične podrobnosti pametnih telefonov in njihove heterogene operacijske sisteme.

Pri forenzični preiskavi mobilnih naprav je potrebno paziti na starost naprave, potrebno je pregledati stanje naprave in v primeru, da je ta zaklenjena, pridobiti ustrezna gesla. Prav tako je potrebno napravo pregledati in opaziti morebitne poškodbe na njej. Forenzična preiskava vsebuje štiri glavna področja analize. To so dnevniški zapisi naprave, omrežni promet, datotečni sistem in delovni pomnilnik. Ključne informacije so pridobljene iz pomnilniških elementov naprav, ki shranjujejo podatke, ti pa nosijo dodatne informacije, katere so odločilne v samem postopku zbiranja dokazov. Ker sami senzorji nimajo trajnega spomina in logike, tj. opravijo določeno funkcionalnost v danem trenutku, s samega senzorja ni mogoče zagotoviti neposrednih informacij. Lahko pa nudijo posredne informacije o tem ali je za zagotovitev nekega podatka res bil prisoten točno določen senzor ali ne. To se ugotavlja s pomočjo karakteristik strojne opreme, na primer leč v senzorju za zajem slike, resoluciji, barvni globini, in nenazadnje tudi načinu transformacije v katerem je zapisana slika.

2.3 Pravice in dolžnosti ponudnikov informacijskih storitev pri forenzični preiskavi

Opravljanje forenzične preiskave je urejeno s strani institucij, ki se ravnajo po kazenskem zakoniku Republike Slovenije. Kazenski zakonik je sprejet s strani državnega zbora, ki potrdi predloge za uvedbo določenega člena kazenskega zakonika. Listina je v tem primeru na voljo v uradnem listu Republike Slovenije, dostopnem tudi na spletu. V okviru preiskave je v pomoč tudi sodelovanje podjetij, ki se morajo ravnati po načelih zapisanih v uradnih listinah države v kateri ponujajo informacijske storitve. V Sloveniji to področje ureja 112. člen zakona o telekomunikacijah[5] - zbiranja in dajanja podatkov, informacij in mnenj ki pravi:

zbiranje in dajanje podatkov, informacij.

1) *Vsi operaterji so na podlagi zahteve dolžni dati agenciji podatke, informacije, finančna poročila oziroma druga poročila, ki jih potrebuje za izpolnjevanje svojih pristojnosti ali za poročanje mednarodnim organizacijam v skladu z mednarodnopravnimi akti, uveljavljenimi v Republiki Sloveniji v obsegu oziroma podrobnostih, kot jih zahteva.*

(2) *Operaterji so dolžni sporočiti osebne in druge podatke le, če tako izrecno določa zakon.*

(3) *Agencija objavlja poročila o stanju iz področja pristojnosti agencije s statističnimi, finančnimi in drugimi podatki organizacij, ki delujejo na teh področjih, pri čemer je v skladu s tem zakonom dolžna varovati poslovno tajnost in druge poslovno občutljive podatke.*

(4) *Agencija je dolžna sporočiti pristojnemu ministru obdelane informacije iz prvega odstavka tega člena.*

(5) *Agencija je dolžna sporočati mednarodnim organizacijam podatke in informacije, ki jih je Republika Slovenija dolžna dajati v skladu s mednarodnopravnimi akti, uveljavljenimi v Republiki Sloveniji.*

(6) *Minister podrobneje predpiše zbiranje in dajanje podatkov.*

Zgornji zakon več ali manj opredeljuje naloge telekomunika-

cijskih podjetij v primeru, ko gre za razkrivanje informacij o uporabnikih v skladu z zakoni. Po drugi strani je pa potrebno zagotavljati zasebnost podatkov. Za ta namen je usposobljena informacijska pooblaščenka, ki v imenu ljudstva skrbi za pravice in zasebnost. To ureja 128. Člen zakona o telekomunikacijah[5] - zaupnost in tajnost telekomunikacij ki velja:

zaupnost in tajnost telekomunikacij.

(1) *Zaupnost in tajnost telekomunikacij se nanaša na vsebino prenesenih sporočil v telekomunikacijskih omrežjih oziroma pri uporabi telekomunikacijskih storitev, na dejstva in okoliščine, v katerih so sporočila prenesena, na dejstvo, ali je ali pa je bil nekdo udeležen pri tem procesu in na dejstva in okoliščine neuspešnih poskusov vzpostavljanja zvez.*

(2) *Vsak operater telekomunikacijskih omrežij ali storitev in vsakdo, ki sodeluje v njihovih dejavnostih, je dolžan varovati zaupnost in tajnost telekomunikacij tudi po koncu dejavnosti, pri kateri jo je bil dolžan varovati.*

(3) *Zavezanci po prejšnjem odstavku tega člena ne smejo sebi ali komu drugemu pridobiti informacij o vsebini, dejstvih ali okoliščinah prenesenih sporočil, ki presegajo najmanjšo potrebno mero, nujno potrebno za opravljanje posameznih telekomunikacijskih storitev, in smejo te informacije uporabljati samo za izvajanje teh telekomunikacijskih storitev ter uresničevanje pogodbenih obveznosti v zvezi z njimi.*

(4) *Če mora operater v skladu s prejšnjim odstavkom tega člena za izvajanje telekomunikacijskih storitev pridobiti informacije o vsebini prenesenih sporočil, posneti ali shraniti preneseno sporočilo, mora o tem ob sklenitvi pogodbe oziroma začetku zagotavljanja telekomunikacijskih storitev obvestiti uporabnika in mu predložiti razloge za to, informacije o vsebini sporočila oziroma sporočilo pa zbrisati takoj, ko je to tehnično izvedljivo in ko to ni več potrebno za izvedbo posamezne storitve.*

(5) *Vse oblike nadzora, posredovanja, prestrezanja in snemanja sporočil, ki se prenašajo z uporabo telekomunikacijskih omrežij ali storitev, so prepovedane, razen v kolikor je to dovoljeno v skladu s prejšnjim odstavkom tega člena in v skladu s 130. členom tega zakona.*

(6) *Naslovnik sporočila lahko sporočilo snema, vendar mora pošiljatelja sporočila o tem obvestiti ali delovanje snemalne naprave prilagoditi tako, da je o njenem delovanju pošiljatelj sporočila obveščen (npr. avtomatski odzivniki), razen pri telekomunikacijskih storitvah, pri katerih je snemanje sporočil pri naslovniku sestavni del telekomunikacijske storitve ali je običajno (npr. faksimilna poročila, elektronska pošta, storitev SMS ...).*

(7) *Ne glede na peti in šesti odstavek tega člena lahko organizacije, ki prejemajo klice na številki 112 in 113, snemajo in zasledujejo te klice zaradi njihove registracije ali zaradi identifikacije zlonamernih klicev.*

(8) *V primeru, da radijski sistem, terminalska ali druga tehnična oprema dobi sporočilo, ki ni namenjeno temu radijskemu sistemu, terminalski ali drugi tehnični opremi, se vsebina teh sporočil ne sme snemati ali uporabljati za katerekoli namene in se mora nemudoma izbrisati ali na drugi način uničiti.*

3. SENZORJI TER OPREDELITEV DOKAZOV

Velika večina sodobnih pametnih telefonov vsebuje strojno opremo, ki omogoča zajem tako analognih kot digitalnih podatkov iz okolja ter hranjenje le teh v pomnilniku naprave. Senzorji so tako učinkovit vir podatkov, ki pridejo v upoštev pri pridobitvi dokaznih gradiv med preiskavami kaznivih dejanj. Tako pripomorejo pri oblikovanju hipotez ter potrditvi oz. ovržbi določenih trditev.

3.1 Opredelitev dokazov

Opredelitev dokazov je definirano kot proces, pri katerem pride do analize podatkov, glede na vir oziroma vsebinske lastnosti. Različni podatki so uvrščeni v kategorije kot so:

- podatki iz sporočil,
- podatki naprave,
- podatki SIM kartice,
- podatki o zgodovini uporabe,
- podatki senzorjev,
- uporabniški podatki.

Ti podatki nudijo informacije ki lahko odgovorijo na zastavljena vprašanja kot so: kdo, kje, kdaj, kaj, zakaj, kako? Ta vprašanja so zastavljena na sodišču, odgovori pa pripomorejo predstavitvi dokazov ter uspešni razrešitvi primera. Različni dokazi ki so predstavljeni so:

Dokazi identitete
Dokazujejo storilce oziroma vpletene osebe v nekem dejanju.
Lokacijski dokazi
Z njihovo pomočjo pridobimo lokacije dokaznih sredstev.
Časovni dokazi
Dokazujejo čas določene akcije, izvedene v okviru preiskovanega dogodka.
Kontekstualni dokazi
Dodajo informacijo o okoliščinah dogodkov in udeležencev, pojasnijo naravo dogodkov.
Motivacijski dokazi
Podatki se lahko uporabijo kot dokaz za motiv preiskovanega dogodka.
Dokazi namena
Dokazujejo namen s katerim je prišlo do dogodka.

Omenjene kategorije podatkov ter različne dokaze lahko povežemo na podlagi korelacij, ki določajo v kolikšni meri določeni dokazi prispevajo k pridobitvi podatkov. Podatki sporočil, naprave ter SIM kartice lahko močno povežemo z dokazi identitet, saj je iz njih možno razbrati identitete oseb. Pri preiskovanju časovnih dogodkov podrobne informacije nudijo podatki iz sporočil pri katerih je zapisan časovni žig naprave ter podatki zgodovine uporabe naprave. Sodobni mobilni telefoni imajo na voljo geolokacijski senzor, preko katerega je možno priti do lokacijskih dokazov. Nekatere kategorije podatkov težko povežemo z določenimi tipi dokazov saj morda ne vsebujejo uporabnih informacij.

3.2 Senzorji

Kljub dejstvu, da je na pametnih mobilnih napravah na voljo veliko različnih senzorjev večina podatkov ne more biti zbrana. Taki podatki so ponavadi občutljivi in težko dostopni, razen v primeru, da jih je naprava beležila. Lažje dostopni so naprimer podatki o GPS lokaciji, ki so zapisani kot metapodatki, vgrajeni v druge datoteke (npr. geotagging) ali shranjeni v kakšni nameščeni aplikaciji kot je Google Maps, Facebook...

Zbiranje lokacijskih podatkov je možno izpeljati tudi brez fizičnega dostopa do mobilne naprave. Ponudniki mobilne telefonije lahko s pomočjo signala oziroma triangulacije med posameznimi baznimi postajami določijo lokacijo naprave. Tako obstajata vsaj 2 možnosti pridobitve dokaza, načini pa se razlikujejo po točnosti, naprava lahko zbira podatke iz enega ali večih senzorjev ki vplivajo na kvaliteto oziroma natančnost podatkov.

Vsak del strojne opreme naprave lahko pri preiskavi vpliva ali zagotavlja podatke na mnogo načinov, naštetih je nekaj transportnih kanalov naprave:

GSM, celično omrežje

Z vidika mobilnih telefonov je GSM (2G) modul skoraj da obvezen del strojne opreme, ki zagotavlja povezljivost naprave v digitalno mobilno celično omrežje z različnimi hitrosti glede na tehnologijo (GPRS, HSDPA, 3G, LTE...) in je prisoten še s časov ko sta bila mobilna infrastruktura in mobilni aparati še v povojih tj. približno začetek devetdesetih let prejšnjega stoletja. Zaradi arhitekture delovanja celičnega omrežja, ima ta določene prednosti in predstavlja dodano vrednost pri opravljanju forenzičnih preiskav. S pomočjo določitve moči signala ter položaja celičnh baznih postaj je mogoče ugotoviti lokacijo mobilne naprave, preko identitete telefona ki je podana z IMSI številko pa je možno vsako napravo tudi identificirati. Mobilni operater lahko ob predložitvi sodnega naloga le te podatke beleži in sledi določeni napravi. Lokacijski podatki s časovnimi značkami kasneje igrajo ključno vlogo pri preiskavi ali rezultatu le te.

Wifi modul

Z rastjo omrežne infrastrukture, interneta, ter napredka strojne opreme so mobilni aparati prerasli v pametne naprave, ki omogočajo dostop do interneta in interakcijo z ostalimi napravami preko operacijskega sistema ter ostalih aplikacij. Te zmožnosti puščajo sledi v obliki podatkov v ostalih napravah kot so usmerjevalniki omrežnega prometa, brezžične dostopovne točke, razne bluetooth naprave itd.

Infrared, NFC, Bluetooth moduli

Preko protokolov manjšega dosega se navadno ne prenaša velike količine podatkov, to vrsto prometa je težko prestreči, saj se prenos podatkov vrši na omejeni razdalji, sledi prenosa pa je možno zaznati na pošiljateljevi in prejemnikovi napravi, kjer ostaja sled o opravljenem prenosu podatka.

GPS

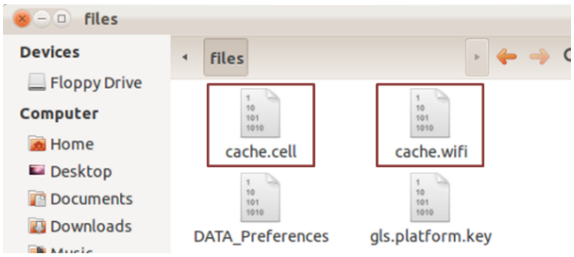
Moderne mobilne naprave imajo vgrajen tudi gps senzor, ki s pomočjo frekvenčnega valovanja, relativnosti in satelitov natančno opiše položaj naprave. Ta informacija se prikaže, shrani v pomnilnik telefona ali pa je na voljo eni izmed aplikacij nameščenih na napravi.

3.3 Uporaba senzorjev v praksi

Z uporabo mobilne naprave nastane situacija različnih dogodkov, ki so pridobljeni s strani senzorjev in uporabniških akcij. S tem nastanejo meta podatki korelacije, te pa nudijo dodatne informacije o nekih okoliščinah. Mobilne naprave so sposobne zajeti sliko, jo obogatiti z metapodatki kot so tip naprave, podatek o uporabniku naprave, geo lokaciji pridobljeni s pomočjo GPS, temperaturi, karakteristiki zajete slike in času zajete slike. Če je bila slika poslana preko omrežja se zagotovi dodatne informacije kot so omrežje na katerega je povezana mobilna naprava, lokacija izvorne naprave, poslani podatki v omrežje, prejemnik poslanih podatkov, lokacija ponorne naprave, ter preko IMSI številke tudi identiteta samega pošiljatelja in prejemnika.

4. PRAKTIČNO DELO

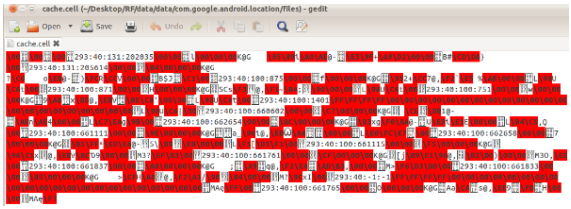
Za praktični del naloge bomo raziskali datoteke, ki vsebujejo podatke o lokacijah na operacijskem sistemu Android. Datoteke, ki vsebujejo lokacije se nahajajo v mapi "/data/data/com.google.android.location/files".



Slika 1: Datoteki s podatki o lokacijah.

Datoteka cache.cell vsebuje mesta lokacij postaj, datoteka cache.wifi pa mesta dostopnih točk do katerih je dostopal uporabnik.

Ko odpremo eno od datotek opazimo, da je ta kodirana v določen format. Ta format se imenuje "java byte stream".



Slika 2: Primer podatkov v datoteki cache.cell.

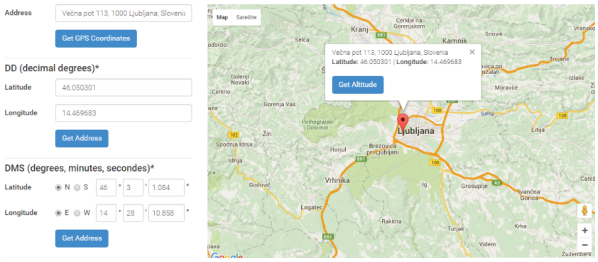
Struktura datoteke je sestavljena tako, da se začne z informacijo o glavi, katera je velika 4 byte. Sledijo podatki o celici oz. njenem fizičnem naslovu (MAC Address) in ostalih informacijah, kot so npr. zaupaje, zemljepisna dolžina in širina ter čas.

Če napišemo program, ki zna prebrati omenjeno datoteko, lahko enostavno preberemo podatke, ki so zapisani v njej. Na sliki 3 so prikazani podatki iz datoteke cache.wifi.

```
00:0e:8f:d9:b8:aa    45    92   46.071577   14.474497   05/10/15 16:55:28 +0000
20:aa:4b:cb:65:cf    50    92   46.049893   14.469308   05/11/15 02:58:33 +0000
02:ca:ff:ee:ba:be    -1     0    0.000000    0.000000   05/11/15 02:58:33 +0000
a2:f3:c1:a7:ec:f4   310    87   46.050013   14.468659   05/11/15 02:58:33 +0000
b0:00:b4:a1:09:c5    75    87   46.049722   14.468789   05/11/15 02:58:33 +0000
00:1d:7e:c6:90:b7    60    92   46.049700   14.467508   05/11/15 02:58:33 +0000
c8:d7:19:38:57:2b    55    92   46.049990   14.469886   05/11/15 02:58:33 +0000
b0:00:b4:a1:09:c1   105    92   46.050302   14.469672   05/11/15 02:58:33 +0000
b0:00:b4:a1:09:c0   105    92   46.050301   14.469683   05/11/15 02:58:33 +0000
a0:f3:c1:3c:20:78    45    92   46.049995   14.469268   05/11/15 02:58:33 +0000
```

Slika 3: Primer prebranih podatkov iz datoteke cache.wifi.

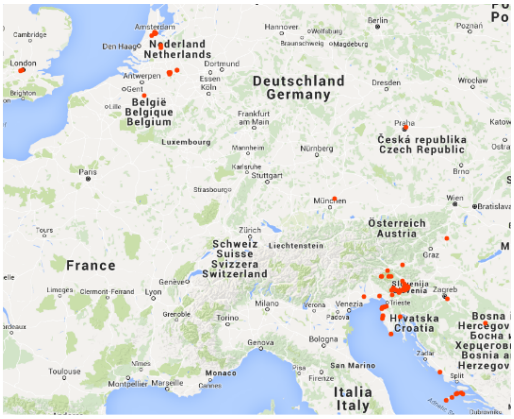
Če eno od zemljepisnih širin in dolžin vpišemo v aplikacijo Google Maps se nam bo izpisal naslov.



Slika 4: Primer lokacije iz datoteke.

Z omenjenim postopkom lahko enostavno pridemo do podatkov o lokaciji na operacijskem sistemu Android.

Ena od možnosti, ki se jo lahko poslužujejo vlade in sodišča je ta, da na podjetje Google pošljejo zahtevo za razkritje uporabniških podatkov. Podjetje Google danes beleži vse vrste informacij o uporabnikih, kot so npr. spletna iskanja, lokacije... Beleženje lokacij za določenega uporabnika storitev je prikazana na spodnji sliki. Iz slike je točno razvidno, kje in kdaj se je določen uporabnik gibal oz. nahajal.



Slika 5: Primer beleženja lokacij za določeno osebo v aplikaciji Google Maps.

Google se sicer zavezuje v spoštovanje zasebnosti in varnosti podatkov, vendar lahko v primeru, da pooblaščene osebe državnega organa zahtevajo razkritje podatkov te tudi razkrijejo.

Spodnja tabela prikazuje za določeno državo število zahtev za pridobitev podatkov, odstotek zahtev, na katere je Google omogočil vpogled v nekaj podatkov in število uporabnikov oz. računov, ki jih je razkril. Tabelo smo omejili le na nekaj držav[7] .

Tabela 1: Število zahtev za pridobitev podatkov od podjetja Google

Država	Število zahtev za pridob. upor. podatkov	% zahtev, kjer so poslali nekaj podatkov	Navedeni računi oz. uporabniki
Slovenija	3	67%	4
Avstrija	33	21%	40
Italija	957	43%	1242
Nemčija	3903	58%	6457
ZDA	12002	78%	31343

5. ZAKLJUČEK

V dokumentu smo predstavili kako se digitalna forenzična preiskava izvaja in nekaj najbolj znanih procesnih modelov za izvajanje forenzične preiskave. Pregledali samo tudi kakšne so pravice in dolžnosti ponudnikov informacijskih storitev ob zahtevi za razkritje podatkov. Opisali smo nekaj možnosti za zajem podatkov različnih senzorjev in uporabo senzorjev v praksi. Za praktično delo smo predstavili kje na Android napravi najdemo podatke o sami lokaciji in kako jih lahko na enostaven način interpretiramo in pregledamo. Kot zanimivost smo dodali še tabelo, ki prikazuje število zahtev za razkritje podatkov s strani podjetja Google.

6. VIRI

[1] Alexios Mylonas, Vasilis Melatiadis, Lilian Mitrou, Dimitris Gritzalis, *Smartphone sensor data as digital evidence*, Computers & Security 38 (2013)

[2] Mark Reith, Clint Carr, Gregg Gunsch *An Examination of Digital Forensic Models*, International Journal of Digital Evidence (2002)

[3] E. Casey, *Digital evidence and computer crime: forensic science, computers and the internet (2nd ed.)*., London:Academic (2004)

[4] Carrier, Spafford *An Event-Based Digital Forensic Investigation Framework*, Information Assurance and Security - CERIAS

[5] (2016) Uradni list RS, zakon o telekomunikacijah. Dostopno na: https://www.uradni-list.si/1/content?id=30706

[6] (2016) Smartphone Forensics: A Proactive Investigation Scheme for Evidence Acquisition Dostopno na: https://pdfs.semanticscholar.org/9b51/4e918a-8ddfbc9d7525c9563dc388f8754d7a.pdf

[7] (2016) Google - Varnost in zasebnost. Dostopno na: https://www.google.com/transparencyreport/userdatarequests/

Network and device forensic analysis of Android social-messaging applications

A paper review

Katja Tuma
Fakulteta za računalništvo in informatiko
Večna pot 113
Ljubljana, Slovenia
kt8269@student.uni-lj.si

Jošt Lajovec
Fakulteta za računalništvo in informatiko
Večna pot 113
Ljubljana, Slovenia
jl7993@student.uni-lj.si

Gašper Volf
Fakulteta za računalništvo in informatiko
Večna pot 113
Ljubljana, Slovenia
gv5308@student.uni-lj.si

ABSTRACT

This paper is a summary of the experiments done to research the security of 20 Android applications in the study under review [16]. Its main focus is set on reconstructing user related data after the use of social-messaging applications. According to their results, even entire message contents were successfully reconstructed from 16 of the 20 applications tested, which shows that security and privacy measures employed by these applications are poor. Besides security it also focuses on forensic analysis of Android smartphones. The paper also describes the methods and results achieved by related work.

Keywords

Network forensics, android forensics, instant messaging, privacy of messaging applications, application security testing

1. INTRODUCTION

Social-messaging applications are nowadays the ultimate means of communication. Applications such as WhatsApp, Viber, Facebook chat, and others, enable a free alternative to communicating via the Internet. As the number of consumers in social media grow, the instant messaging (IM) is changing the way people interact with each other. As a result, a gap in security and privacy measures needs to be addressed by social-media software authors [10]. Furthermore, as D. S. Terzi et al. [14] point out, it is not only very difficult to store big data and analyse them with traditional applications, but it also has challenging privacy and security problems. However, the interactive and instant nature of these applications made them an attractive choice for malicious cyber activities such as phishing [17]. The paper in review, is trying to stress the significance of realising the security gap and taking advantage of the possibility of retrieving user-related data, and using this information to prevent and discover criminal activity. The following sections briefly describe the area of related work, paper objectives, methodology, and discussion.

2. RELATED WORK

2.1 Areas of related work

Mobile phones and their applications are easily involved in various criminal events, such as theft, money laundering, scams, child pornography or malicious software. Even before the development of smartphones, call history and text messages were obtained from mobile phones. They both represented crucial evidence for digital forensics. Today most of the phones are smartphones that contain all sorts of data, such as text messages, emails, photographs, videos and many others. Therefore, a lot of evidence in different forms can be found there. In comparison with other potential sources of digital evidence, smartphones are mostly located right next to their owners. This increases the chance of the device being at the crime scene and consequently implies that there might be some useful information and digital evidence on the device [11].

Most smartphones use Android as their operating system. There are many different smartphones manufacturers and even more brands and models, which define procedures for obtaining data from their devices. Many such techniques are further described in [3].

Logical acquisition of data can be performed with backup utilities, which are designed not to modify the device or the system software. Nevertheless, in order to be able to get full access to the devices secondary storage and be able to use a tool like dd, some kind of rootkit is needed, which modifies the devices system partition, as described in the study [11]. Such rootkits are most easily acquired by sourcing directly from the hacking community. Consequently, it is better to avoid or at least minimise the usage of such tools. As proposed by Vidas at al. [15], the Android recovery partition might be more safely overwritten within a safe forensic boot environment. As a result, physical acquisition of the remaining partitions is possible, and the system partition is not modified by the rootkit. The access to the device's secondary storage is then obtained by rebooting the device into a new and modified recovery mode with the needed software to perform a physical acquisition. This works similarly to using a boot CD to perform acquisition on a personal computer. Physical acquisition is from a forensic perspective better than logical, because it gives access to deleted

files which were not yet overwritten. Despite this, logical acquisition can give significant evidence and continues to be under study.

There are many studies in digital forensics literature about social networks and mobile applications for instant messaging. Husain and Sridhar's study on the iPhone [9] focused on platforms which were originally implemented for personal computers (PCs) as a standalone application or via the web. Reust described techniques for the examination of artifacts from AOL Instant Messenger [12]. Dickson analysed traces of interactions between users of AOL Instant Messenger [6], Yahoo! Messenger [8] and MSN Messenger [7]. There have been studies on other installed instant messaging applications and instant messaging features of social networking websites such as Facebook [1]. Instant messaging platforms working on personal computers have migrated to the smartphone. Since then digital forensics are investigating activity traces left by these applications on mobile devices. Newer social networking and instant messaging applications were developed primarily for the smartphone. One of the most used messaging application is WhatsApp. With results from Anglano's recent study [2], an analyst will be able to reconstruct the chronology of the messages, list of contacts and even information such as date and time of adding a specific contact. Smartphones have become the main targets of cyber attacks, which should be no surprise given the fact that almost every person owns one. Smartphones are also easy to target because of the way applications are distributed, that is community driven. The official applications stores on Android and iPhone do not allow malware, but there are a number of unofficial stores. This makes distributing malware simpler than ever before. In the study [5] they worked on inter-application communication and they found 1414 vulnerabilities in the top 50 paid and free application on what was then called Android Market, now known as Google Play. Smartphone applications for instant messaging were also shown by Schrittwieser et al. to be vulnerable to account hijacking, spoofing, unrequested SMS, enumeration and other attacks in the study [13]. Given that smarthphones contain lots of personal information those attacks pose risks to privacy.

Security flaws often make digital forensics job to recover digital evidence easier but on the other hand they result in higher number of cybercrime incidents targeting mobile devices. Walnycky's work complements existing literature by employing network forensics as well as device forensics to provide more holistic view of what evidence may be obtained from messaging applications on Android devices. Potential privacy issues that arise from weak security implementation were also discovered.

2.2 Facebook's instant messaging service

Al Mutawa et al. in their study [1] highlight the importance of Facebook's instant messaging service as a potential source of evidence in an investigation. In their experiment they created two fictional Facebook users and a set of unique strings which would be used as the instant messaging conversations and would be easy to identify during the examination process. They created three sets of unique strings, each one to be tested on a single web browser - Internet Explorer, Firefox and Chrome. They proved that instant messages exchanged during a Facebook Chat session in the web browser are cached and stored on the computer's hard disk. The locations of stored messages vary depending on the web browser used to produce the communication. The highest number of artifacts of the chat session can be found on the hard disk when using Internet Explorer.

3. SUMMARY OF PAPER

The following sections briefly outline the content of the aforementioned research paper. We find, that some aspects of our readings have a greater importance, hence we focused on further elaborating on the paper objectives, methodology used in their approach and discussion of results. The paper is well structured and contains a section on related work, methodology, experimental results, discussion and future work. The main purpose of writing the paper, as we understand, was to conduct hands-on experiments in order to test the extent to which measures for privacy and security insurance are applied in different Android messaging applications. Walnycky et al. [16], also show that the unfortunate lack of security measures employed by instant messaging applications, leads to the features leaving traces behind, allowing suspect data to be partly or fully reconstructed. The reconstructed data can sometimes be of vital importance to an investigation and serve as evidence in court. The authors were able to reconstruct the following types of data: passwords, screenshots taken by applications, sent pictures, video, audio, messages, profile pictures and more. They further elaborate on the vast usage of mobile applications and show how important is the knowledge of intercepting such data in criminal situations. What is more, the relative simplicity of intercepting and reconstructing such data is being put forward, in order to demonstrate the possibilities in the field of digital forensics and security. The following section includes some notes on the latest research on privacy and security measures taken by social-messaging applications.

3.1 Paper objectives

Their main objective was to illustrate the potential for acquiring digital evidence from the mobile device, data in transit, and data stored on servers. Furthermore, the paper contains a section on related work, where they discuss the popularity of smartphones, Android and Apple (iOS) products, and how digital evidence obtained in such a way, was collected in the past. In order to present the most important aspects of digital mobile forensics, the importance of the SIM card related information is mentioned. Furthermore, the paper lists the possible usage of tools such as rootkit and the difference between physical and logical data acquisition. Just to clarify, physical data acquisition is for example acquiring all the data written on the disk, while logical data acquisition would be only gathering the data within the scope of the native file system. Physical data acquisition is generally preferred over logical data acquisition, yet due to case-dependent limitations, it is unfortunately sometimes impossible to gather all data [4]. What is more, early work on instant messaging examined platforms which were originally released for the personal computer either as a standalone application or via the web. Therefore, computer forensics techniques described in the literature from early stages of research focus on IM applications such as AOL, Yahoo! Messenger, etc. As these messaging platforms moved to smartphones with time, so did the area of research and digital forensics community started to investigate the activity traces left behind by

applications on mobile devices. In addition to these imports from the PC, instant messaging and social networking applications were developed primarily for mobile devices. Given the popularity of smartphones, it is not surprising that they have become targets for cyber attacks. While some of this are enabling the digital forensic community to gather data and obtain evidence, most of them are exploited and used for harm in cybercrime.

3.2 Methodology

Walnycky et al. selected 20 instant messaging applications from the Google Play store, based on their popularity, namely, Tinder, Wickr, Snapchat, BBM, WhatsApp, Viber, Instagram, Okcupid, ooVoo, Tango, Kik, Nimbuzz, MeetMe, MessageMe, TextMe, Grindr, HeyWire, Hike, textPlus and Facebook Messenger. The common trait of all selected applications is the support for one on one communication, the functionality of which was under study. They performed network forensics to examine the network traffic captured by sending and receiving messages, using multiple features of applications (eg. sending photos, or videos as well). Testing was done in a controlled lab environment to reduce possible network variability. They also performed a forensic examination on the Android device itself, to compare the captured content with the original information (eg. if there is any image compression performed, etc.). Figure 1 shows a list of the tested applications in the order they were tested, their version numbers, and the features they support.

3.2.1 Network analysis setup

Following is the description of the testing environment they used to produce experimental results. They used a HTC One M8 model, as well as iPad 2 as shown in Figure 2. The Android device was the target of their examination, while the iPad only served as a communication partner to exchange messages with the HTC phone. They used a Windows 7 computer with WiFi and an Ethernet connection to the Internet to set up a wireless access point. This PC unit was used to capture the traffic sent over the WiFi in both directions. Both devices were, during the testing, connected to the newly setup wireless access point, with the use of Windows 7 Virtual WiFi Miniport Adapter feature. Wireshark was used to capture and save the network traffic, which was further analysed in the process.

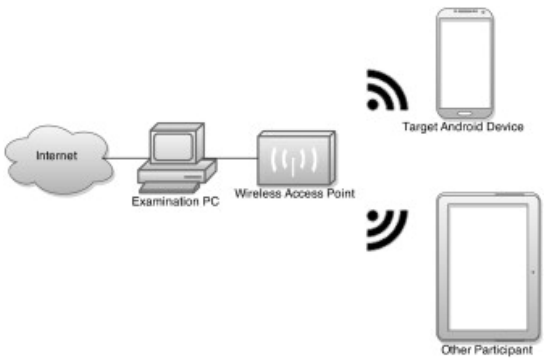


Figure 2: Experimental network setup.

3.2.2 User activity

Once the network was set up, they performed a series of actions, varying from application to application, depending on their functionalities and capabilities, in order to examine all the messaging types supported. The aforementioned table 1, includes the list of varieties they tried. When a single activity trace was found for an evidence type it was documented.

3.2.3 Data storage experimental setup

After all the network analysis was completed, they used different tools to forensically investigate the data storage on the phone. They used Microsystemation's XRY to perform a logical acquisition of the Android device. They cross-validated the results using an open source alternative, Helium backup extractor and sqlite database browser to view the contents of .db files on the phone. Evidence documented for each application was found in a single .db file that contained the chat logs and/or user information. The information obtained from the database analysis phase was then compared to the captured (un)encrypted content by Wireshark. For further details on all technologies used, proceed to reading the full section on methodology from the original paper [16].

3.3 Experimental results

Four out of twenty applications were tested with no resulting vulnerabilities (Tinder, Wicks, Snapchat and BBM). They are using HTTPS encryption with SSL certificates. Nevertheless, they were able to intercept some kind of unencrypted data with the rest of listed applications. According to the authors of the original paper, the lack of end-to-end encryption may be due to resources available to developers or because companies deem this data as non-privacy invasive. In four cases, they were able to capture the actual messages sent (MessageMe, MeetMe, ooVoo and Okcupid), where they were even able to capture the content of messages in plain text with testing Okcupid. They were able to also reconstruct multimedia content, including images, videos, locations, sketches and audio (Instagram, ooVoo, Tango, Nimbuzz, MessageMe, textPlus, TextMe, Viber, HeyWire, Grindr, and Facebook Messenger). Captured profile images from Tango are shown in Figure 3.

Applications	Capabilities	Performed activity	Network traffic traces	Data storage traces	Server traces
WhatsApp (2.11)	Text chat Audio, video, image, location, and V-card sharing	Sent/Received message Sent/Received voice note Sent/Received image Sent video Sent V-card Sent/Received GPS location	V-Card Location (Sent)		
Viber (4.3.0.712)	Text chat Voice call Audio, video, image, sketch, and location sharing	Sent/Received message Sent/Received sketch Sent/Received image Sent/Received voice note Received voice call Sent/Received GPS location	Images (Received) Sketches (Received) Video (Received) Location (Sent/Received)		Images, Video, Sketches
Instagram (6.3.1)	Text chat Video and image sharing	Sent/Received image	Images (Sent/Received)		Images
Okcupid (3.4.6)	Text chat	Sent/Received message Sent/Received image	Text (Sent)		
ooVoo (2.2.1)	Text chat Voice call Video call Video and image sharing	Sent/Received message Sent/Received image	Text (Sent/Received) Images (Sent/Received)	Chat Log	Images
Tango (3.8.95706)	Text chat Voice call Video call Audio, Video, and image sharing	Sent/Received message Sent/Received image	Images (Sent/Received)		Videos
Kik (7.3.0)	Text chat Video, image, and sketch sharing	Sent/Received message Sent/Received image Sent/Received sketch	Sketches (Sent)	Chat Log	
Nimbuzz (3.1.1)	Text chat Voice call Audio, video, image, and location sharing	Sent/Received message Sent/Received image Sent/Received GPS Location Sent/Received Video	Location (Sent) Images (Sent/Received) Video (Sent)	Plain Text Password Chat Log	
MeetMe (8.6.1)	Text chat Image sharing	Sent/Received message Sent/Received image	Text (Sent/Received)	Chat Log	
MessageMe (1.7.3)	Text chat Audio, video, image, sketch, and location sharing	Sent/Received message Sent/Received image Sent/Received sketch Sent/Received GPS Location Sent/Received Video Sent/Received Music	Location (Sent/Received) Text (Sent/Received) Images (Sent/Received) Sketches (Received) Music (Sent)		Videos
TextMe (2.5.2)	Text chat Voice call Video call Video and image sharing	Sent/Received message Sent/Received image Sent Dropbox file Received video	Location (Sent/Received) Images (Received)	Plain Text Password Chat Log	
Grindr (2.1.1)	Text Chat Image and location sharing	Sent/Received message Sent/Received image	Images (Sent)		Images
HeyWire (4.5.10)	Text chat Voice call Audio, image, and location sharing	Sent/Received message Sent/Received image Sent/Received GPS Location	Location (Sent) Images (Received)	Chat Log	Images
Hike (3.1.0)	Text chat Voice call Audio, video, image, location, and V-Card sharing	Sent/Received message Sent/Received image Sent/Received GPS Location	Location (Sent)	Chat Log	
textPlus (5.9.8)	Text chat Voice call Audio and image sharing	Sent/Received message Sent/Received image	Images (Sent/Received)	App Taken Screenshots, Chat Log	Images
Facebook Messenger (25.0.0.17.14)	Text chat Voice call Audio, video, image, location, and stickers	Sent/Received message Sent/Received image Sent/Received video Sent/Received audio Sent/Received GPS location Sent/Received stickers	Images (Sent/Received) Video Thumbnails (Received)		Images, Video Thumbnails

Figure 1: Application capabilities, actions, and traces.

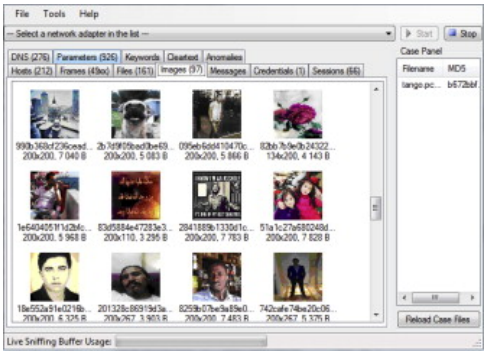


Figure 3: Captured profile images from Tango.

Furthermore, during the network traffic testing, they were able to recover links to applications servers, where the content was still existent and unencrypted. This allows for any investigator with access to these links to download the data. What is more, with the process of logical acquisition of a mobile device, they were able to sometimes see the remains of missing files, which they were able to reconstruct info full chat logs in plain text. It was also possible to retrieve some other user-sensitive data, such as passwords, user names, e-mails, phone numbers and such, from examination of the Android device. Sensitive user data obtained from TextMe is shown in Figure 4. Again, for further details, please review the contents of the paper [16].

	id	key	value
	Filter	Filter	Filter
1	24	database.migration	3
2	47	user.username	unhcfregdroid
3	48	user.email	[redacted]@live.com
4	49	user.phone	+1203[redacted]
5	50	user.sms_number	+1914[redacted]
6	51	user.gender	m
7	52	user.birthday	1992/07/22
8	53	user.password	gin[redacted]
9	54	user.password_hashed	false
10	55	user.userId	47763570

Figure 4: TextMe database with user data in plaintext.

3.4 Provided discussion and suggestions for future work

The paper clearly shows that the majority of social-messaging applications used today lack in security and privacy measures features and therefore have significant problems with ensuring private use. Nevertheless, the investigative significance of security gaps is discussed to be actually more than welcome. Even though, they claimed to have tried notifying the developers of the problem at stake, no response was noticed. As a result, the authors call upon the community to further engage in continuous testing of new versions and similar applications to encourage the awareness of the issues described. They claim, there is very little effort required to reconstruct a similar testing environment. After they completed their research, they even developed their own open source tool "Datapp"to further automate the process and make it easy for users to be able to conduct these test themselves.

4. PROPOSED EXPERIMENT

After a group discussion on the matter of what would be a valid similar experiment, we decided that it would actually be sensible to test their newly developed framework. From the end-user point of view, it is more important that an open source, easy to use software is available, rather than reconstructing a different network based lab environment in order to obtain similar results. We also feel like it would be useful to conduct similar tests on newest versions of afore mentioned IM applications. In case of conducting the experiment below are listed some basic system requirements obtained from their web page:

1. Windows 7 installed on the PC,
2. Ethernet interface and a wireless interface on the PC,
3. Windows user account needs to allow running applications in administrative mode.

5. CONCLUSION

After having read and discussed the topics, we are able to conclude that it has been shown time and again, that the lack of privacy and security measures in social-messaging applications is very real, and should be regarded as a bigger concern not only from consumers, but also from their creators. What is even more alarming is that most consumers (end application users), that are endangered of falling under attack, still don't seem to realise it. The convenience of free available applications, and wide usage of them, pushes people info obliviously agreeing to terms and conditions that are violating their right to privacy. Looking at the matter from the digital forensics perspective, we found that ironically this security gap, or in other words, the inadequate design of social-messaging applications, allows the digital forensics community to develop powerful tools for acquiring data to support evidence in criminal investigations.

6. REFERENCES

[1] N. Al Mutawa, I. Al Awadhi, I. Baggili, and A. Marrington. Forensic artifacts of facebook's instant messaging service. In *Internet Technology and Secured*

Transactions (ICITST), 2011 International Conference for, pages 771–776. IEEE, 2011.

[2] C. Anglano. Forensic analysis of whatsapp messenger on android smartphones. *Digital Investigation*, 11(3):201–213, 2014.

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

[4] E. Casey. *Digital evidence and computer crime: Forensic science, computers, and the internet*. Academic press, 2011.

[5] E. Chin, A. P. Felt, K. Greenwood, and D. Wagner. Analyzing inter-application communication in android. In *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services*, MobiSys '11, pages 239–252, New York, NY, USA, 2011. ACM.

[6] M. Dickson. An examination into aol instant messenger 5.5 contact identification. *digital investigation*, 3(4):227–237, 2006.

[7] M. Dickson. An examination into msn messenger 7.5 contact identification. *digital investigation*, 3(2):79–83, 2006.

[8] M. Dickson. An examination into yahoo messenger 7.0 contact identification. *digital investigation*, 3(3):159–165, 2006.

[9] M. I. Husain and R. Sridhar. iforensics: forensic analysis of instant messaging on smart phones. In *Digital forensics and cyber crime*, pages 9–18. Springer, 2009.

[10] S. Kumar, K. Saravanakumar, and K. Deepa. On privacy and security in social media–a comprehensive study. *Procedia Computer Science*, 78:114–119, 2016.

[11] J. Lessard and G. Kessler. Android forensics: Simplifying cell phone examinations. 2010.

[12] J. Reust. Case study: Aol instant messenger trace evidence. *digital investigation*, 3(4):238–243, 2006.

[13] S. Schrittwieser, P. Frühwirt, P. Kieseberg, M. Leithner, M. Mulazzani, M. Huber, and E. R. Weippl. Guess who's texting you? evaluating the security of smartphone messaging applications. In *NDSS*, 2012.

[14] D. S. Terzi, R. Terzi, and S. Sagiroglu. A survey on security and privacy issues in big data. In *2015 10th International Conference for Internet Technology and Secured Transactions (ICITST)*, pages 202–207. IEEE, 2015.

[15] T. Vidas, D. Votipka, and N. Christin. All your droid are belong to us: A survey of current android attacks. In *WOOT*, pages 81–90, 2011.

[16] D. Walnycky, I. Baggili, A. Marrington, J. Moore, and F. Breitinger. Network and device forensic analysis of android social-messaging applications. *Digital Investigation*, 14:S77–S84, 2015.

[17] T. Y. Yang, A. Dehghantanha, K.-K. R. Choo, and Z. Muda. Windows instant messaging app forensics: Facebook and skype as case studies. *PloS one*, 11(3):e0150300, 2016.

Del III

Forenzika slik

Določanje modelov digitalnih kamer s podporo neznanih modelov

[Povzetek članka: Camera Model Identification With Unknown Models [6]]

Blaž Kostanjšek
Fakulteta za Računalništvo in Informatiko
Večna pot 113
Ljubljana, Slovenija
blaz.kostanjsek@gmail.com

Tilen Kavčič
Fakulteta za Računalništvo in Informatiko
Večna pot 113
Ljubljana, Slovenija
tk5866@student.uni-lj.si

Peter Rot
Fakulteta za Računalništvo in Informatiko
Večna pot 113
Ljubljana, Slovenija
peter.rot93@gmail.com

POVZETEK

Določanje modela digitalne kamere na podlagi lastnostih izluščenih iz fotografije v digitalni forenziki igra pomembno vlogo. Žal imajo vse obstoječe tehnike, ki temeljijo na večrazredni klasifikaciji skupen problem določanja neznanih modelov. Nekatere fotografije so namreč zajete z digitalno kamero, ki sistemu ni vnaprej znana zato obstoječe tehnike takšne modele napačno klasificirajo v najbližji ciljni razred. Avtorji članka v ta namen predlagajo novo tehniko imenovano SCIU (Source Camera Identification with Unknown models), ki je sposobna tako zaznati ali gre za neznan model kamere kot tudi razlikovati med znanimi modeli kamer. Predlagana tehnika je sestavljena iz treh delov: 1) detekcija neznanih modelov, 2) razširitev neznanih modelov, ter 3) klasifikacija v enega izmed $K+1$ razredov. Detekcija neznanih modelov uporablja metodo k-najbližjih sosedov (KNN) z namenom prepoznavanja primerov neznanih modelov v neoznačeni množici fotografij. Razširitev neznanih modelov nato množico neznanih modelov še dodatno razširi. V zadnji fazi pa združimo množico neznanih modelov (1 razred) ter množico znanih modelov (K razredov), ter le-to uporabimo za učenje na $(K+1)$-razrednem klasifikatorju. Avtorji članka razvijejo tudi tehniko optimizacije parametrov tehnike SCIU. Razvita tehnika je nato preizkušena na množici slik Dresden, kjer avtorji pokažejo dejansko učinkovitost razvite tehnike v primerjavi s tehnikami večrazrednega klasificiranja s podpornimi vektorji (MSVM), binarnega klasificiranja s podpornimi vektorji (BSVM), kombiniranih klasifikacijskih tehnik (CCF) ter klasificiranja DBC (decision boundary carving). Dodatno skušamo avtorjevo delo ter raziskave ponoviti, ter preveriti ali se dobljene rezultati ujemajo s tistimi o katerih poročajo avtorji članka.

Ključne besede

določanje modela kamere, neznani modeli, strojno učenje, digitalna forenzika

1. UVOD

Z vse večjim porastom tehnologije digitalnih fotografij in na drugi strani popularnostjo ter dostopnostjo digitalnih kamer iz dneva v dan nastaja vedno več digitalnih fotografij. Digitalne fotografije se danes s strani mnogih organizacij uporabljajo tudi kot dokazi pri sprejemanju odločitev. V vseh teh primerih pa je prav določanje modela kamere, s katero je bila digitalna fotografija posneta, ena izmed ključnih zahtev. Uporaba tehnike določanja modela kamere je na primer za sodišče pomembna pri določanju lastnika kamere, s katero je bila digitalno fotografija posneta. Prav tako igra pomembno vlogo pri določanju pravega avtorja posnete digitalne fotografije, v primeru kršitve avtorskih pravic. Primerov uporabe takšne tehnike je kar nekaj, v članku pa avtor izhaja predvsem iz problematike, ki se dotika področja digitalne forenzike.

V splošnem lahko problem določanja modela digitalne kamere rešujemo na tri načine:

1. glede na metapodatke vsebovane v sami sliki,
2. glede na morebitno prisotnost vodnega žiga, ter
3. glede na lastnosti izluščene iz samih fotografij.

V prvem načinu zgolj izluščimo tako imenovane metapodatke vsebovane v sliki, ki vsebujejo informacije o znamki in modelu kamere, datumu in času ko je bila fotografija posneta, lokaciji, kjer je bila fotografija posneta, objektivu in nastavitvah, ki so bile uporabljene, itd. Ta način je med drugim tudi najenostavnejši, saj so nam vse informacije že dane in jih je potrebno zgolj prebrati. Žal pa ima ta način tudi pomanjkljivosti: metapodatke slik je namreč možno spreminjati na precej enostaven način, kar nas lahko privede do napačnih zaključkov, saj obstaja možnost, da so bile prave informacije o kameri namerno spremenjene.

Drugi način skuša model digitalne kamere določiti na podlagi vodnega žiga, ki ga nekatere kamere vgradijo v fotografijo. Problem tega načina je, da vodnega žiga ne podpirajo vse digitalne kamere, saj mora biti vodni žig vgrajen v fotografijo takoj, ko je le ta posneta, kar dodatno zakomplicira izdelavo digitalnih kamer, ter posledično dviguje cene digitalnih kamer.

Tretji način, ki je v članku podrobneje predstavljen, pa predstavlja tehniko, kateri se v zadnjem času posveča vse več pozornosti, ter temelji na razlikovanju slik glede na določene lastnosti. Pri tej tehniki sprva iz fotografij izluščimo lastnosti, ki so posledica samega procesa zajema digitalne fotografije s pomočjo digitalnega senzorja. Vsak digitalni senzor za zajem fotografij ima namreč neke fizične lastnosti, ki se odražajo na digitalni fotografiji. Poleg tega pa v procesu zajemanja digitalnih fotografij nastopa še programski del, ki zajeto sliko pretvori v določen slikovni format in pri tem uporablja tudi morebitno kompresijo. Tudi tu programska oprema različnih proizvajalcev različno vpliva na zajeto digitalno fotografijo. Ko iz množice fotografij izluščimo izbrane lastnosti, le-te uporabimo kot učne primere pri učenju večrazredne klasifikacije, na primer klasifikacije s podpornimi vektorji - support vector machine (SVM). Kasneje, ko želimo za novo fotografijo določiti model, uporabimo naučen klasifikator za uvrstitev fotografije v enega izmed modelov prisotnih v učni množici.

Problem pri takšnih vrstah večrazredne klasifikacije predstavljajo neznani modeli digitalnih kamer. Dejstvo namreč je, da vseh možnih modelov digitalnih kamer nimamo podanih vnaprej. Tudi če večrazredni klasifikator naučimo na vseh možnih modelih digitalnih kamer, bo s prihodom novega modela kamere tak klasifikator nepopoln. Ker pa novi modeli digitalnih kamer prihajajo na tržišče vse pogosteje, takšne vrste večrazredne klasifikacije postajajo vse manj natančne. Problem neznanih modelov namreč močno vpliva na klasifikacijsko točnost modela, saj v primeru, da v model kot testni primer vstopa fotografija neznanega modela, naučeni model skuša fotografijo uvrstiti v enega izmed modelov, ki so bili prisotni v učni množici - kar je seveda napačna uvrstitev.

Ravno zaradi te problematike neznanih modelov digitalnih kamer avtorji članka predlagajo novo tehniko večrazredne klasifikacije, imenovano Source Camera Identification with Unknown models (SCIU), ki uspešno rešuje ta problem. Tehnika sestoji iz treh glavnih delov. V prvem delu s pomočjo klasifikacije k-najbližjih sosedov (KNN) razvijemo metodo za prepoznavanje neznanih modelov, kjer iz neoznačene učne množice - kjer razredi niso poznani - določimo kateri primeri pripadajo neznanim modelom. V drugem delu še dodatno razširimo množico neznanih modelov z vpeljavo metode za razširitev neznanih modelov. V zadnjem delu pa problem rešujemo s pomočjo (K+1)-razredne klasifikacije. Primere fotografij neznanih modelov (1-razred) ter primere fotografij znanih razredov (K-razredov) združimo, ter se na njih učimo večrazredne klasifikacije SVM.

2. SORODNO DELO

Identifikacija modela kamere preko lastnosti naprej pridobi vse lastnosti, ki jih za sabo pusti proces za pridobitev slike. Nato pridobljene lastnosti posreduje algoritmu za nadzorovano učenje. Mnogo raziskovalnih del je bilo posvečeno temu, vendar se jih zelo malo posveča za prepoznava neznanih modelov.

2.1 Lastnosti za identifikacijo modela kamere

Lastnosti kamere lahko razdelimo v dve kategoriji:

1. Strojni artefakti
2. Prstni odtisi programske opreme

Med strojne artefakte spadajo šum[8], radialna popačenje leče[9], kromatska aberacija[12] in prah na senzorju [2]. Med programsko opremo sodijo lastnosti slike[7] in artefakti, ki jih pustijo barvno filtri.

Kharrazi et al.[7] je v svojem delu demonstriral kako izstopajo artefakti po uporabi barvnega filtra. Predlagal je 34 lastnosti posnetka preko katerih poteka identifikacija. Geradts el al.[3] se je ukvarjal s pokvarjenimi slikovnimi pikami na senzorju, ki pustijo za unikatni prstni odtis. Dirik et al.[2] je pokazal, da se lahko na podobni način kot s slikovnimi pikami uporabiš obliko in pozicijo delcev prahu na senzorju za identifikacijo. Choi et al.[9] je modele identificiral s pomočjo radialnega popačenja slik. Van et al. [12] je predstavil metodo za identifikacijo preko kromatke aberacije. Kromatska aberacija povzroči, da se določeni spektri vidne svetlobe pod različnimi konci, kar povzroči popačenje barv. Lukaš et al. [8] je prvi predstavil metodo za pridobitev vrednost odzivnosti senzorja (PRNU). Najprej so izničili šum slikam in s tem pridobili odzivnost senzorja. Nato so izračunali povprečno vrednost odzivnosti na množici slik. Na koncu so normalizirali korekcijski koeficient med PRNU in PRNU vrednosti kamere. Sutcu el al. [11] je izboljšal Lukaševo metodo.

Slika 1: Kromatske aberacije.

2.2 Problem neznanih modela

Po pridobitvi lastnosti obravnavamo identifikacijo modelov kamer kot K-razredni klasifikacijski problem (K je število znanih modelov). Nato se problem reši z več razrednimi klasifikatorji kot na primer "multi-class SVM (MSVM)". Pri

učenju "multi-class SVM"je potrebno paziti, da je učna množica sestavljanja iz znanih modelov kamer. MSVM vrne razred v katero spada testna slika. Če je prisotna slika neznanega modela, bo sliko napačno uvrstil. Le par ljudi je vložilo v razvoj reševanja problema neznanih modelov kamer. Gloe [4] je raziskoval dva načina zasnovana na enojnem razredu SVM in binarnem SVM. V enojnem razredu SVM učimo s pomočjo množice znanih modelov. Z njim preverimo ali je bila slika narejena z znanim modelom. Vendar SVM potrebuje veliko učno množico za učinkovitost. Medtem pa je binarni SVM zasnovan na primerjalnim večrazrednim SVM. Neznani modeli se obravnavajo kot razredi. Pri tem se identifikacija naredi preko (K + 1) razrednim klasifikatorjem. Učenje znanih in neznanih modelov poteka iz množice znanih modelov. Učna množica pri tem nima neznanih modelov, ki bi zmanjševala hitrost BSVM. Eno razredni SVM je naučen prepoznati posamezni model kamere. Če ne zna umestiti slike v razred znanih modelov, je slika umeščena v razred neznanih modelov. Če slika spada v več razredov, uporabimo več razredni SVM za identifikacijo znanega modela. Binarni SVM lahko torej naučimo, da znane modele uporabimo kot pozitivne primere ostale znane pa kot negativne primere. Pri upoštevanju negativnih primerih lahko nastavimo mejo odločanje v prid pozitivnih primerov. S tem zmanjšamo možnost, da se v prihodnosti srečamo z negativno pozitivnimi rešitvami. Vendar meja odločanja je lahko zaradi pomanjkanja informacij o neznanih slabo zastavljena.

Trenutne rešitve se ukvarjajo s prepoznavo neznanih modelov brez uporabe informacij o neznanih modelih. Motivacija dela je uporaba informacij o neznanem za iskanje neznanega in predlog nove sheme za identifikacijo neznanih modelov.

3. PREDLOG NOVE TEHNIKE

3.1 Predstavitev SCIU

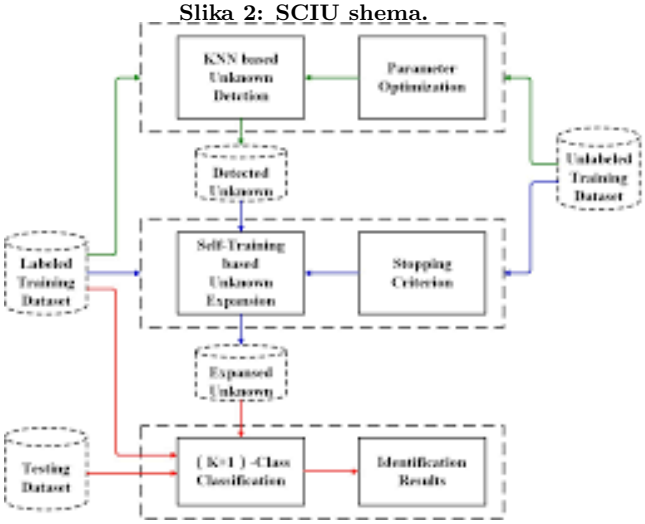
Članek predlaga metodo z imenom SCIU (Source Camera Identification with Unknown models). Cilja te metode sta ugotoviti identiteto slik, za katere nimamo znanega modela, ter razlikovanje slik pri katerih je model znan. Pri tem je potrebno omeniti, da je ta članek osredotočen na klasificiranje v več razredov. Zato izpostavlja eno izmed šibkosti klasičnih algoritmov za tovrstno klasificiranje, kot je na primer SVM (Support vector machine). Ta šibkost je slaba klasifikacija v razred novih modelov, ki v učni množici niso bili prisotni, se pa pojavijo v testni množici. Težko je namreč pričakovati, da bodo vsi proizvajalci aparatov za zajem fotografij v enotno podatkovno zbirko prispevali slike vseh modelov aparatov. Metoda je bila v članku preizkušana na podatkovni zbirki slik z imenom Dresden (Dresden image collection. SCIU deluje v naslednjih treh fazah:

1. Detekcija neznanih modelov (Unknown detection)
2. Razširitev neznanih modelov (Unknown expansion)
3. Klasifikacija v enega izmed K+1 razredov ((K +1)-class classification)

Podatkovna zbirka je razdeljena na tri dele, in sicer:

1. Neoznačene slike, ki služijo v fazi učenja (ne vemo katerim napravam za zajem slik pripadajo)
2. Označene slike, ki služijo v fazi učenja vemo, katerim fotoaparatom (napravam) pripadajo
3. Testna množica (naključno izbrane slike iz obeh zgornjih skupin)

Slika 2: SCIU shema.



V prvi fazi metode SCIU moramo zaznati, katere slike nimajo znanega modela, kar pomeni, da nimajo preglednice. To naredimo s pomočjo algoritma za iskanje K-najbližjih sosedov (K-nearest neighbours). V drugi fazi, ki se imenuje razširitev neznanih modelov, razširimo množico vzorcev, ki nimajo znanega modela s pomočjo samo-učeče se strategijo (self-training strategy). V tretji fazi pa gre za klasifikacijo z K+1 razredom, pri čemer je K število znanih modelov, dodatni razred pa pripada neznanim modelom. Članek poleg opisa teh treh faz predlaga tudi metodo za optimizacijo parametrov, proučevali pa so tudi ustavitveni kriterij pri neznanih modelih. Avtorji trdijo tudi, da je SCIO boljši od SVM ter binarnega SVM.

3.2 Detekcija neznanih modelov

Za detekcijo neznanih modelov je predlagan algoritem K-najbližjih sosedov (v nadaljevanju KNN). KNN deluje na sledeči način. Za vsako neoznačeno sliko, recimo ji I, moramo preveriti, če njenih K najbližjih sosedov vsebuje sliko znanega modela. Če ga vsebujejo, potem za I-to sliko lahko sklepamo, da je bila slika zajeta z modelom, ki je bil definiran pri K-tem sosedu. Če idejo matematično formuliramo, imamo označeno podatkovno zbirko $\mathbb{P}$, za katero velja $\mathbb{P} = U_{i=1}^{p} P_i$, kjer je P_i zbirka slik, ki so bile zajete z napravo w_i. Imamo tudi učno neoznačeno podatkovno zbirko $\mathbb{Q}$. Najprej naredimo unijo množic $\mathbb{P}$ in $\mathbb{Q}$, poimenujmo jo $\mathbb{T}$. Potem moramo za vsako sliko I, ki pripada množici $\mathbb{Q}$ poiskati K najbližjih sosetov v $\mathbb{T}$. Razdaljo merimo s pomočjo evklidske razdalje. I bomo označili kot neznan model, če v množici K-tih najbližjih sosedov ne bomo našli slike, ki pripada $\mathbb{P}$. Algoritem je prikazan na sliki 3.

3.3 Optimizacija parametra K

Parameter K moramo optimizirati, saj drugače algoritem ni dovolj natančen. V članku je predstavljena tudi teoretična analiza, ki pokaže, da je verjetnost, da bomo sliko, ki

Slika 3: KNN detekcija neznanih modelov.

Algorithm 1 KNN Based Unknown Detection

Input : Labelled training dataset : $\mathbb{P} = \bigcup_{i=1}^{p} P_i$.
Unlabelled training dataset : $\mathbb{Q}$.
The parameter : K
Output: Image set of unknown models : $\mathbb{U}$

$\mathbb{U} \leftarrow \emptyset$;
//Combine the labelled training dataset and unlabelled training dataset.
$\mathbb{T} \leftarrow \mathbb{P} \bigcup \mathbb{Q}$;.
foreach Image $I \in \mathbb{Q}$ **do**
  //Find the K nearest samples of I in $\mathbb{T}$ with the Euclidean distance.
  $\mathbb{N}^I \leftarrow \mathsf{KNN}(I, \mathbb{T}, K, \text{Euclidean distance})$;
  if $\mathbb{N}^I \bigcap \mathbb{P} == \emptyset$ **then**
    // If $\mathbb{N}^I$ doesn't contain any image in $\mathbb{P}$,I is labelled as unknown.
    $\mathbb{U} \leftarrow \mathbb{U} \bigcup \{I\}$;
  end
end
return $\mathbb{U}$.

v resnici pripada neznanemu modelu, napačno klasificirali v razred znanega modela, zelo majhna. To pomeni, da ima metoda SCIU teoretično visoko natančnost. Pokazano je, da če K nastavimo med 1 in 10, potem je $P(w_1, ...w_i, ...w_W|I)$ med 0.039 in 0.004. Za optimizacijo parametra K uporabimo metriki TPR (True positive rate) in FPR (False positive rate). TPR je definiran kot število pravilno zaznati slik, ki spadajo v razred neznanih slik, ulomljeno s številom vseh slik, ki spadajo neznanim slikam. FPR je definiran kot število zaznanih slik, ki jim napačno pripišemo, da imajo neznan model, ulomljeno z številom vseh slik, za katere je model znan. Eksperiment pokaže, da je natančnost klasifikacije močno odvisna od parametra K. Pri spreminjanju K iz 1 na 10 se TPR klasificirani slik (med neznane modele) spremeni iz 94% na 27%, medtem ko se FPR spremeni iz 52% na 0%. Izkaže se, da se tako TPR kot FPR zmanjšujeta, ko K narašča. Iz tega sklepamo, da je optimalen K tisti K, pri katerem je FPR približno enak nič.

Zavedati se moramo, da v praksi ne moremo izmeriti FPR množice $\mathbb{Q}$, zato moramo FPR množice $\mathbb{Q}$ aproksimirati z (v nadaljevanju opisano) metodo. Na začetku iz označene množice slik $\mathbb{P}$ vzamemo majhno podmnožico (10% primerkov) in jo tretiramo, kot da so primerki iz te podmnožice neoznačeni (zbrišemo jim labelo). To podmnožico označimo z $\Delta\mathbb{P}$. Nato z unijo združimo množici $\Delta\mathbb{P}$ in $\mathbb{Q}$ in to unijo poimenujemo $\mathbb{Q}'$. Nato izračunamo FPR na množici $\Delta\mathbb{P}$ in s tem dobimo aproksimacijo FPR na množici $\mathbb{Q}$. Članek nam z grafom pokaže tudi, da je aproksimacija FPR zelo dobra. Algoritem je prikazan na sliki 4.

3.4 Razširitev neznanih modelov

Množico neznanih modelov moramo razširiti, saj bomo lahko z metodo tako odkrili več neznanih modelov. To naredimo

Slika 4: Optimizacija parametra K.

Algorithm 2 Parameter Optimization for Unknown Detection

Input : Labelled training dataset : $\mathbb{P}$.
Unlabelled training dataset : $\mathbb{Q}$.
The FPR′ threshold : $\mathsf{T_{fpr}}$ (default 0.5%).
Output: The optimal K.

$\Delta\mathbb{P} \leftarrow$ randomly selected 10% of images from $\mathbb{P}$;
$\mathbb{P}' \leftarrow \mathbb{P} - \Delta\mathbb{P}$;
$\mathbb{Q}' \leftarrow \mathbb{Q} + \Delta\mathbb{P}$;
$\mathbb{T}' \leftarrow \mathbb{P}' \bigcup \mathbb{Q}'$;
for $K \leftarrow 1$ *to* $K_{\max}$ **do**
  $\mathbb{U}' \leftarrow \emptyset$;
  foreach Image $I \in \Delta\mathbb{P}$ **do**
    //Find the K nearest samples of I in $\mathbb{T}'$ with the Euclidean distance.
    $\mathbb{N}^I \leftarrow \mathsf{KNN}(I, \mathbb{T}', K, \text{Euclidean distance})$;
    if $\mathbb{N}^I \bigcap \mathbb{P}' == \emptyset$ **then**
      // If $\mathbb{N}^I$ doesn't contain any image in $\mathbb{P}'$,I is labelled as unknown.
      $\mathbb{U}' \leftarrow \mathbb{U}' \bigcup \{I\}$;
    end
  end
  $\mathsf{FPR'_K} \leftarrow \frac{|\mathbb{U}'|}{|\Delta\mathbb{P}|}$;
  if $\mathsf{FPR'_K} \leq \mathsf{T_{fpr}}$ **then**
    $\mathsf{K_{opt}} \leftarrow \mathsf{K}$;
    break;
  end
end
$\mathsf{K_{opt}} \leftarrow \mathsf{K_{max}}$;
return $\mathsf{K_{opt}}$.

s pomočjo samo-učeče se metode (self-training procecure), ki temelji na ideji bootstrapinga. Bootstraping pri metodah strojnega učenja po navadi izboljša klasifikacijsko natančnost in zmanjšuje problem pretiranega prileganja podatkov. To dela na tak način, da v proce1s učenja vključuje neoznačene podatke. Vhodni parametri v algoritem so označena učna množica $\mathbb{P}$, neoznačena množica $\mathbb{Q}$ ter množica $\mathbb{U}$, ki jo sestavljajo elementi, ki so v fazi prepoznavanja neznanih modelov prepoznane kot elementi neznanega modela. Algoritem je prikazan na sliki 5.

3.5 Klasifikacija v (K+1) razred

Za razliko od K razredno identifikacijo SCIU uporablja (K+1) razredno identifikacijo. Pri tem neoznačenim slikam nameni poseben razred. Število vseh znanih modelov naprav za zajem slik pa je število K. Avtorji članka so za problem večrazrednega klasificiranja uporabili algoritem tako imenovano metodo podpornih vektorjev (Support vector machine ali krajše SVM), saj se le-ta velikokrat izkaže za zelo uspešno metodo.

Slika 5: Algoritem za razširitev neznanih modelov.

Algorithm 3 Self-Training Based Unknown Expansion

Input : Labelled training dataset : $\mathbb{P}$.
Unlabelled training dataset: $\mathbb{Q}$.
Extracted unknown of unknown detection : $\mathbb{U}$. The DIR threshold : $\mathsf{T}_{\mathsf{dir}}$ (default 0.5%).
Output: The expansed unknown : $\bar{\mathbb{U}}$.

$\bar{\mathbb{U}}_0 \leftarrow \mathbb{U}$;
$\bar{\mathbb{Q}}_0 \leftarrow \mathbb{Q} - \mathbb{U}$;
for $s \leftarrow 1$ *to* S **do**
  $\mathbb{T}_s \leftarrow \mathbb{P} \bigcup \bar{\mathbb{U}}_{s-1}$;
  //Regard $\mathbb{T}_s$ as training dataset to train a multi-class SVM $\mathsf{C_s}$.
  $\mathsf{C_s} \leftarrow \mathsf{TrainSVM}(\mathbb{T}_\mathsf{s})$;
  //Use $\mathsf{C_s}$ to classify images in $\bar{\mathbb{Q}}_{s-1}$.
  $\mathsf{L}(\bar{\mathbb{Q}}_{\mathsf{s}-1}) \leftarrow \mathsf{SVMPredict}(\mathsf{C_s}, \bar{\mathbb{Q}}_{\mathsf{s}-1})$;
  //$\Delta\mathbb{U}_s$ is the set of images labelled as unknown by C_s, 0 denotes the label of the class for unknown models.
  $\Delta\mathbb{U}_s \leftarrow \{I | I \in \bar{\mathbb{Q}}_{s-1} \wedge L(I) == 0\}$;
  $\bar{\mathbb{U}}_s \leftarrow \bar{\mathbb{U}}_{s-1} \bigcup \Delta\mathbb{U}_s$;
  $\bar{\mathbb{Q}}_s \leftarrow \bar{\mathbb{Q}}_{s-1} - \Delta\mathbb{U}_s$;
  $\mathsf{DIR_s} \leftarrow \frac{|\Delta\mathbb{U}_s|}{|\bar{\mathbb{U}}_{s-1}|}$;
  if $\mathsf{DIR_s} \leq \mathsf{T}_{\mathsf{dir}}$ **then**
    return $\bar{\mathbb{U}}_s$;
  end
end
return $\bar{\mathbb{U}}_s$.

4. IMPLEMENTACIJA TEHNIKE SCIU

Avtorji članka izvorne kode predlagane SCIU sheme niso javno objavili, zato smo se odločili za lastno implementacijo tehnike, glede na opis podan v članku. Delo smo v osnovi razdelili na tri dele. Namen prvega dela je pridobiti nabor fotografij nad katerimi so avtorji članka opravili raziskave, ter iz njih izluščiti častnosti, kot je to predlagano v članku. V drugem delu sledi preverjanje klasifikacijskih točnosti različnih, že obstoječih tehnik, ki so omenjene v članku. Namen zadnjega dela pa je, na podlagi podanih algoritmov pripraviti delujočo shemo SCIU, vendar dejanske implementacije nismo izvedli zaradi pomanjkanja lastnosti.

4.1 Pridobivanje značilk

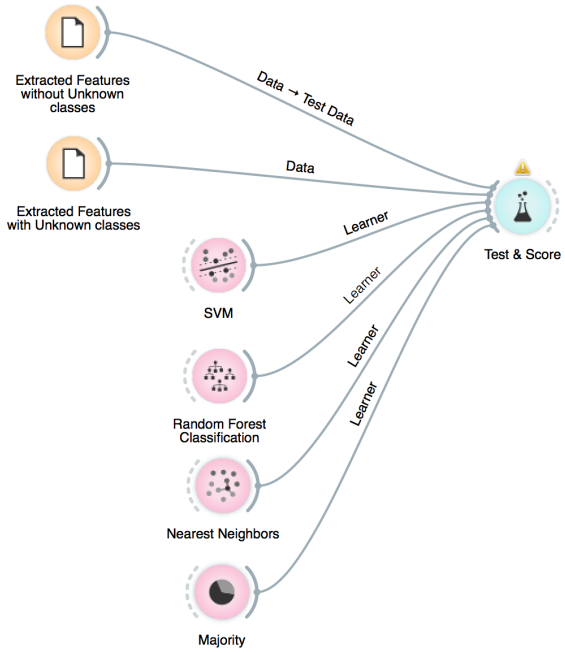
Izbor lastnosti smo izvedli na prosto dostopni množici fotografij Dresden, tako kot avtorji članka. Avtorji sicer navajajo, da so si pri samem pridobivanju lastnosti pomagali z izvorno kodo člankov v katerih so predlagane in podrobneje opisane lastnosti [7] [10]. Glede na to, da v nobenem izmed teh člankov avtorji izvorne kode za luščenje lastnosti niso javno objavili, smo po predlaganih opisih razvili skripto (glej prilogo A) za pridobivanje tistih lastnosti, ki so bile v člankih dovolj dobro opisane. Žal je bilo takšnih lastnosti zgolj 6, vendar smo vseeno nadaljevali s preverjanjem učinkovitosti obstoječih tehnik, pri čemer smo se zavedali, da utegnejo biti rezultati slabši, kot jih navajajo avtorji.

4.2 Preverjanje učinkovitosti obstoječih tehnik

Po pridobljenih lastnosti smo želeli preveriti ali pri že obstoječih tehnik klasifikacijska točnost res pada z večanjem števila neznanih modelov v sistemu. Naša prva predpostavka je bila, da utegnejo biti klasifikacijske točnosti slabše od tistih, ki jih navaja avtor, vendar pa naj bi sam trend kazal v podobno smer.

Za preverjanje učinkovitosti različnih klasifikacijskih tehnik smo uporabili grafično orodje za podatkovno rudarjenje Orange [1]. Sestavili smo shemo prikazano na sliki 6. Pri tem smo uporabili dva nabora podatkov. V prvem podatkovnem naboru so za vse primere definirani njihovi pripadajoči razredi, v drugem primeru pa smo postopoma odstranjevali po en razred, do skupno 20 neznanih razredov. Prvi podatkovni nabor je bil tako namenjen testiranju, drugi pa učenju. Ker pa orodje Orange ne ponuja vseh tehnik, ki so jih avtorji preverjali v članku, smo uporabili nekaj izmed omenjenih, poleg tega pa dodali še nekaj svojih. Tako smo preverjali tehnike: večrazredno klasificiranje s podpornimi vektorji (SVM), klasificiranje z naključnimi gozdovi (RF), klasificiranje s k-najbližjimi sosedi (KNN) ter zgolj za primerjavo uporabili še napovedovanje večinskega razreda (Majority).

Slika 6: Shema implementacije testiranja obstoječih klasifikacijskih tehnik v okolju Orange.



5. TESTIRANJE IN REZULTATI

5.1 Pregled avtorjevih rezultatov testiranj

5.1.1 Testna množica in nastavitve

Avtorji so empirične preizkuse izvajali na Dresden-ovi izbirki slik. Izbirka je bila ustvarjena v namen razvoja in preizkušanja forenzičnih tehnik za digitalne kamere. Sestavljena je

Tabela 1: Pregled različnih naprav za zajem fotografije.

No.	Model	Size	Alias	No.	Model	Size	Alias	No.	Model	Size	Alias
1	Agfa_DC – 504	169	A1	10	FujiFilm_FinePixJ50	630	F1	19	Pentax_OptioW60	192	P3
2	Agfa_DC – 733s	281	A2	11	Kodak_M1063	2391	K1	20	Praktica_DCZ5.9	1019	PR1
3	Agfa_DC – 830i	363	A3	12	Nikon_CoolPixS710	925	N1	21	Ricoh_GX100	854	R1
4	Agfa_Sensor505 – x	172	A4	13	Nikon_D200	752	N2	22	Rollei_RCP – 7325XS	589	RO1
5	Agfa_Sensor530s	372	A5	14	Nikon_D70	369	N3	23	Samsung_L74wide	686	S1
6	Canon_Ixus55	224	C1	15	Nikon_D70s	367	N4	24	Samsung_NV15	645	S2
7	Canon_Ixus70	567	C2	16	Olympus_mju_1050SW	1040	O1	25	Sony_DSC – H50	541	SO1
8	Canon_PowerShotA640	188	C3	17	Panasonic_DMC – FZ50	931	P1	26	Sony_DSC – T77	725	SO2
9	Casio_EX – Z150	925	C4	18	Pentax_OptioA40	638	P2	27	Sony_DSC – W170	405	SO3

Tabela 2: Pregled neznanih modelov.

No.	Unknown Model	No.	Unknown Model
1	C2	14	SO3
2	S1	15	C1
3	P3	16	C3
4	PR1	17	K1
5	P2	18	N3
6	R1	19	A3
7	O1	20	A2
8	N4	21	F1
9	N2	22	S2
10	RO1	23	P1
11	N1	24	SO1
12	A5	25	C4
13	SO2	26	A4

iz slik naravnega in urbanega območja ter slik zunanje in notranjega okolja. V tabeli 1 je pregled nad modeli in števili slik, ki jih ima posamezni model.

Avtorji so uporabili model za pridobitev lastnosti slik, ki ga je predlagal Kharrazi et al. [7]. V model spadajo naslednje tehnike:

1. Povprečna vrednost točk (3 lastnosti)
2. RGB parna soodvisnost (3 lastnosti)
3. Težiščna razporeditev sosedov (3 lastnosti)
4. Razmerje RGB parov energije (3 lastnosti)
5. Spektralna statistična analiza domene (9 lastnosti)
6. Kvaliteta slike (13 lastnosti)

Izvorna koda za pridobitev lastnosti je bila pridobljena preko Kharrazi et al.[7]. Posamezna slika je bila predstavljena kot vektor z 34 dimenzijami.

Število posameznih slik za vsak model je navedeno v tabeli 1. Avtorji so naključno razdelili množico slik za posamezni model v 3 podmnožice enakih velikosti. Tri podmnožice so bile uporabljene kot označena trening množica, neoznačena trening množica in testna množica. Avtorji so simulirali probleme neznanih modelov tako da so naključno izbrane znane modele označili kot neznane. Število neznanih modelov je vidno v tabeli 2. Avtorji so za reševanje SVM uporabili LIBSVM. Prav tako so v svojih poskusih uporabili Gaussovo krivuljo za optimizacijo parametrov SVM-ja.

5.1.2 Metrika vrednotenja rezultatov

Avtorji poročajo, da so uporabili naslednje metrike za vrednotenje rezultatov:

- Splošna natančnost (OACC) je meritev kjer je gledamo razmerje med pravilno identificiranimi slikami s številom vseh identificiranih slik.

$$\texttt{OACC} = \frac{\texttt{\# pravilno identificiranih slik}}{\texttt{\# vseh identificiranih slik}}$$

- Natančnost poznanih (KACC) se uporablja za preverjanje pripadnosti slik k znanemu modelu. Definiran je kot razmerje med številom pravilno identificiranimi slikami, ki pripadajo znanemu model in številom slik, ki ne moremo identificirati.

$$\texttt{KACC} = \frac{\texttt{\# pravilno identificiranih slik znanih modelov}}{\texttt{\# slik neznanih modelov}}$$

- Natačnost nepoznanih (UACC) se uporablja za preverjanje razpoznavanja neznanih slik. Definirana je kot razmerje med številom pravilno identificiranih slik, ki pripadajo neznanemu modelu in številom slik, ki so identificirane kot neznane.

$$\texttt{UACC} = \frac{\texttt{\# pravilno identificiranih slik neznanih modelov}}{\texttt{\# slik neznanih modelov}}$$

- F-meritev [5] je se uporablja pri merjenju zmogljivosti identificiranja modela w_i, ki je kombinacija med natančnostjo in priklicom.

$$\texttt{f-meritev}^i = 2 \cdot \frac{\texttt{natačnost}^i \cdot \texttt{priklic}^i}{\texttt{natačnost}^i + \texttt{priklic}^i}$$

  Kjer je $\texttt{natančnost}^i$ razmerje med številom pravilno identificiranih slik modela w^i in številom slik, ki spadajo k modelu w^i.

5.1.3 Vpliv neznanih modelov

Avtorji so nato pokazi kako vplivajo neznani modeli na MSVM shemo. MSVM-jeva učna množica je vsebovala slike znanih modelov. Testa množica je pa vsebovala kombinacijo znanih in neznanih modelov. Na grafu Fig. 7 so predstavljene OACC vrednosti pri različnem številu neznanih modelov v testi množici. Če pogledamo celo črto na grafu, opazimo da nam MSVM pristop pri uvedbi neznanih modelov povzroči hiter padec klasifikacijske natančnosti. Iz grafa je razvidno, da ima testna množica z znanimi modeli klasifikacijsko natančnost 92%. Ko pa predstavimo 20 neznanih modelov v testno množico, pa nam natančnost pade na 23%. To se zgodi, ker nam MSVM pristop klasificira neznane modele kot znane.

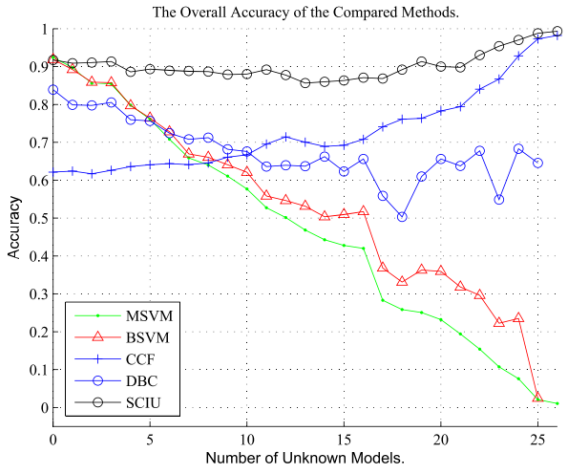
5.1.4 *Primerjava metod*

Avtorji so nato primerjali MSVM, BSVM, CCF in DBC z predlagano SCIU shemo.

Natančnost identifikacije

Fig .7,Fig .8,Fig .9 prikazujejo podatke o OACC, KACC, UACC vseh 5 metod. Fig 10, 11 and 12 prikazujejo F-meritev vseh 5 metod pri različnem številu neznanih modelov. Iz grafov je razvidno, da je predlagana metoda SCIU boljša kot vse ostale. Druga najboljša je CCF metoda. Povprečna razlika med SCIU in CCF metodo je 18%. MSVM nam na najslabšo natančnost, BSVM je za malenkost boljša kot MSVM in DBC je boljša kot BSVM. MSVM metoda ne razloči med neznanimi in znanimi modeli. Medtem ko, BSVM in DBC do določene mere upoštevata ostale znane modele vseeno delata aproksimacijo za neznane modele. Vendar DBC poskuša izboljšati aproksimacijo z omejevanjem izbire pri SVM-ju.

Slika 7: Primerjava končnih klasifikacijskih točnosti različnih metod.



Posledično DBC doseže boljšo natančnost kot BSVM. Vendar je treba pomniti, da CCF, DBC in BSVM ne izkoriščajo informacij o neznanih modelih. Posledično je njihova zanesljivost omejena. Predlagana metoda SCIU izkorišča ravno te informacije preko identifikacije neznanega.

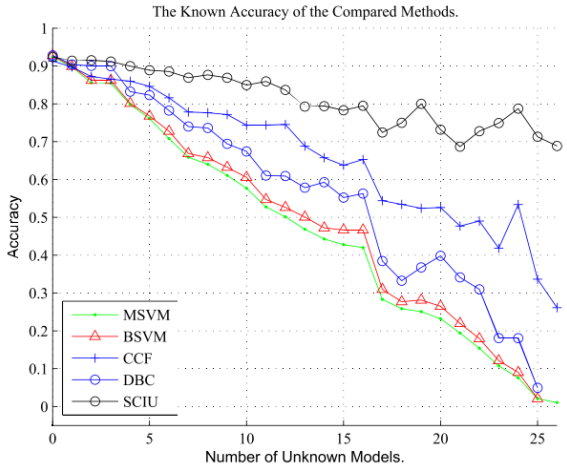
1. **OACC:** Vrednost OACC se pri MSVM, BSVM in DBC zmanjša sorazmerno s število neznanih modelov. CCF metoda pri povečanju števila neznanih modelov dobi boljšo OACC vrednost. OACC vrednost se pri SCIU sorazmerno ohranja s številom neznanih modelov.

2. **KACC:** Vrednost KACC pri vseh petih metodah pada sorazmerno z številom neznanih modelov. Vendar BSVM, CCF, DBC in SCIU uporabljajo drugačne prijeme za zmanjševanje negativnega vpliva neznanih modelov. Ker SCIU shema dobro odkrije neznane informacije, prehiti vse ostale metode.

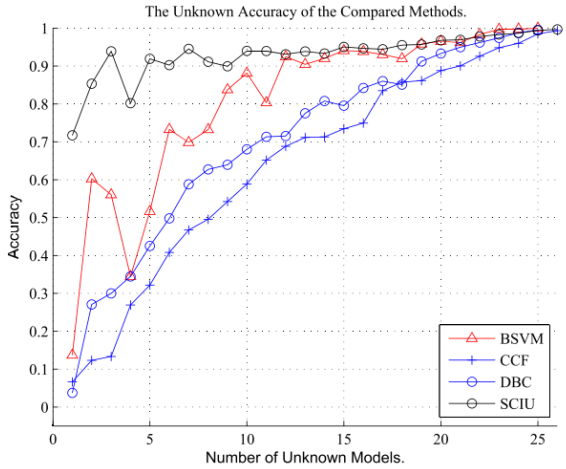
3. **UACC** Vrednost UACC se pri metodam BSVM, CCF, DBC in SCIU narašča sorazmerno s številom neznanih modelov. SCIU tudi tukaj prevladuje.

4. **F-meritev** Iz Fig. 10,11 in 12 vidimo, da metoda SCIU je učinkovitejša kot pa ostale metode. Za 5,10 in 20 neznanih modelov ima SCIU F-vrednost 84%, 88% in 93%. Metodi SCIU spodleti pri zelo podobnih modelih.

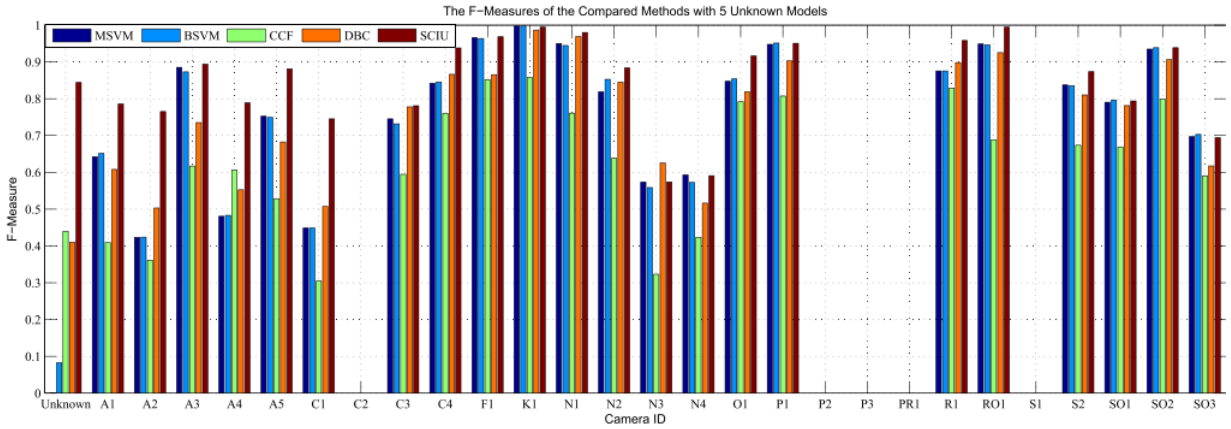
Slika 8: Primerjava klasifikacijskih točnosti različnih metod za znane modele.



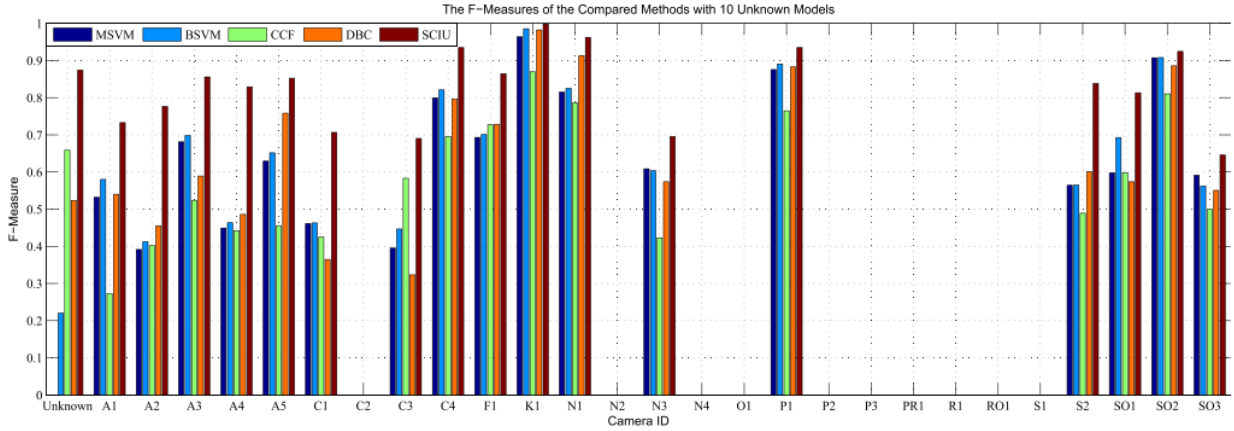
Slika 9: Primerjava klasifikacijskih točnosti različnih metod za neznane modele.



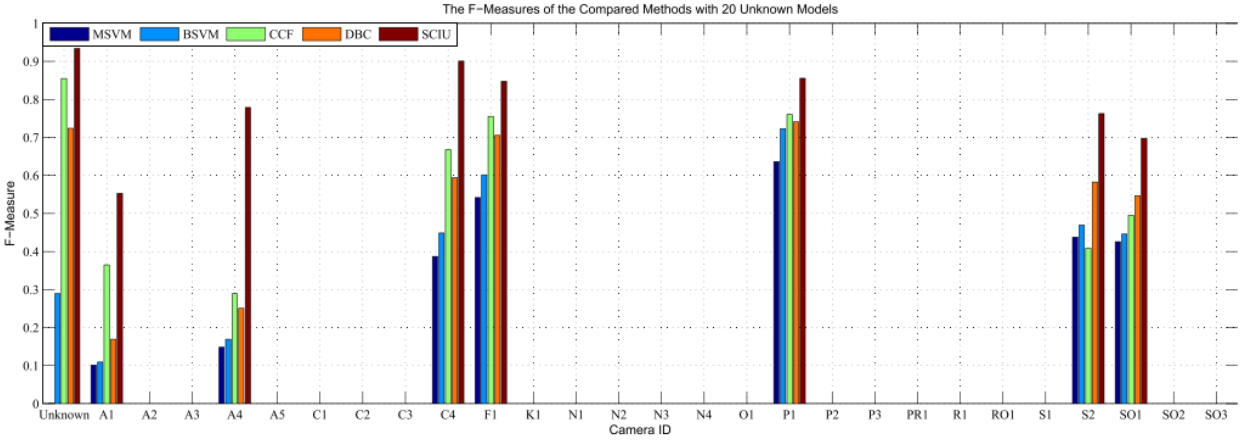
Slika 10: Primerjava F-ocen različnih metod, ki imajo 5 neznanih modelov.



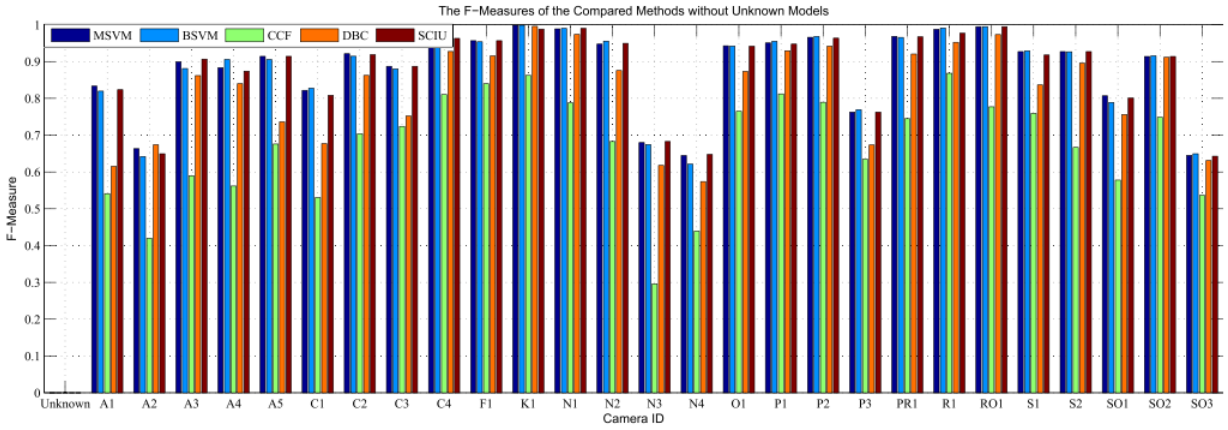
Slika 11: Primerjava F-ocen različnih metod, ki imajo 10 neznanih modelov.



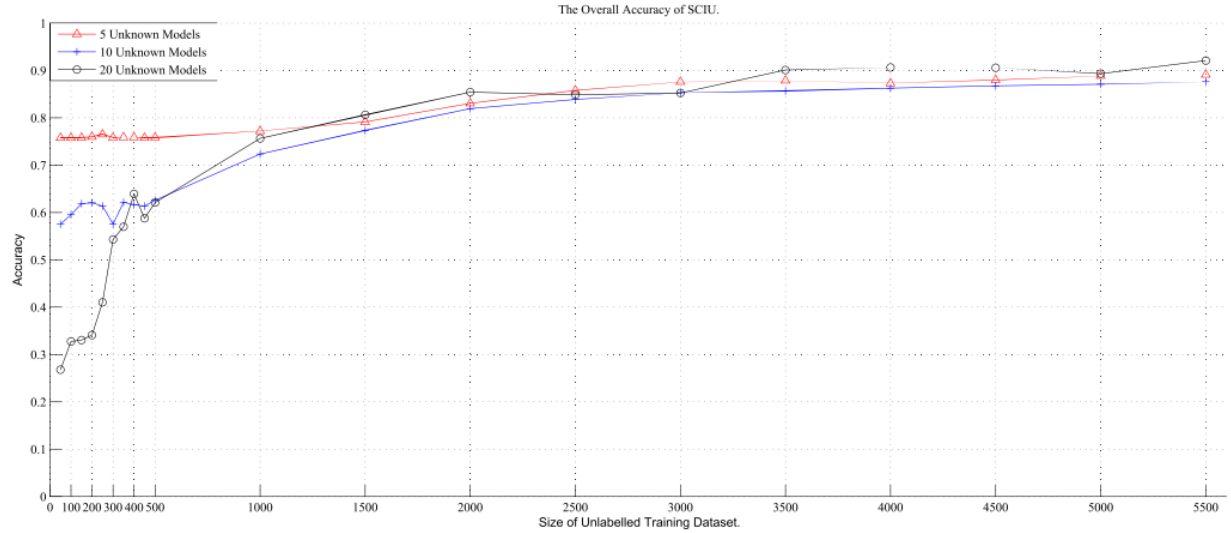
Slika 12: Primerjava F-ocen različnih metod, ki imajo 20 neznanih modelov.



Slika 13: Primerjava F-ocen različnih metod, ki nimajo neznanega modela.



Slika 14: Končna klasifikacijska točnost SCIU.



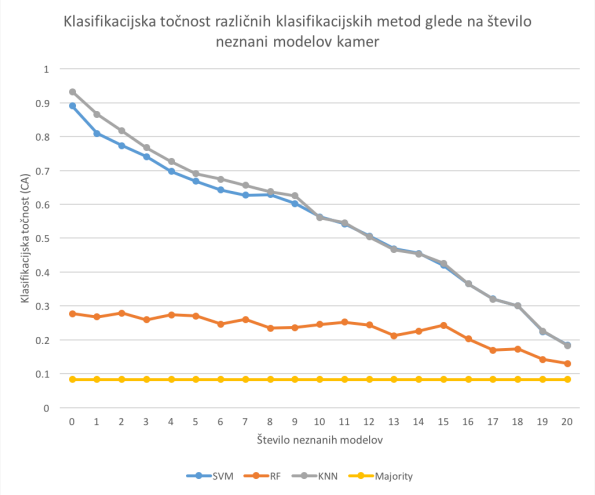
5.2 Rezultati testiranj na lastni implementaciji

Na lastni implementaciji smo preverili avtorjeve rezultate o klasifikacijskih točnostih obstoječih tehnik. Rezultati testiranja so prikazani na sliki 15. Kot smo pričakovali, so klasifikacijske točnosti nižje od tistih, o katerih poročajo avtorji, saj smo uporabili zgolj 6 lastnosti. Ne glede na to pa opazimo, da je trend padanja klasifikacijske točnosti z večanjem števila neznanih modelov enak kot v članku. Jasno je torej, da obstoječe tehnike delujejo precej slabše, če se v sistemu pojavijo neznani modeli digitalnih kamer.

Preverili smo še dve tehniki, katerih avtorji v članku niso preverjali, k-najbližjih sosedov (KNN) ter naključne gozdove (RF). Tehnika KNN dosega podobne rezultate kot tehnika SVM, zanimivo pa je, da se v tem primeru tehnika naključnih gozdov ne obnese.

Zgolj za primerjavo smo prikazali še podatke o uvrščanju v večinski razred (Majority), kjer vedno napovemo razred, ki je v podatkih najpogostejši. Kot pričakovano se klasifikacijska točnost te tehnike ne spreminja in je enaka deležu fotografij najpogostejšega razreda med vsemi fotografijami.

Slika 15: Klasifikacijska točnost različnih klasifikacijskih metod glede na število neznanih modelov digitalnih kamer.



6. ZAKLJUčEK

Avtorji izpostavljajo, da trenutne metode za identifikacijo modelov kamer niso zanesljive zaradi neupoštevanja neznanih informacij o modelih. Preizkuse so opravili na odprti bazi slik Dresden. Predlagana metoda SCIU pokrije pomanjkljivost s pomočjo dodatnih metod, ki pravilno odkrijejo neznane modele. Posledično je SCIU trenutno najzanesljivejša metoda za tovrsten preiskave.

7. REFERENCES

[1] Orange - data mining tool. `http://orange.biolab.si/`.

[2] A. E. Dirik, H. T. Sencar, and N. Memon. Digital single lens reflex camera identification from traces of sensor dust. *Information Forensics and Security, IEEE Transactions on*, 3(3):539–552, 2008.

[3] Z. J. Geradts, J. Bijhold, M. Kieft, K. Kurosawa, K. Kuroki, and N. Saitoh. Methods for identification of images acquired with digital cameras. In *Enabling Technologies for Law Enforcement*, pages 505–512. International Society for Optics and Photonics, 2001.

[4] T. Gloe. Feature-based forensic camera model identification. In *Transactions on Data Hiding and Multimedia Security VIII*, pages 42–62. Springer, 2012.

[5] G. Hripcsak and A. S. Rothschild. Agreement, the f-measure, and reliability in information retrieval. *Journal of the American Medical Informatics Association*, 12(3):296–298, 2005.

[6] Y. Huang, J. Zhang, and H. Huang. Camera model identification with unknown models. *Information Forensics and Security, IEEE Transactions on*, 10(12):2692–2704, 2015.

[7] M. Kharrazi, H. T. Sencar, and N. Memon. Blind source camera identification. In *Image Processing, 2004. ICIP'04. 2004 International Conference on*, volume 1, pages 709–712. IEEE, 2004.

[8] J. Lukas, J. Fridrich, and M. Goljan. Digital camera identification from sensor pattern noise. *Information Forensics and Security, IEEE Transactions on*, 1(2):205–214, 2006.

[9] K. San Choi, E. Y. Lam, and K. K. Wong. Automatic source camera identification using the intrinsic lens radial distortion. *Optics express*, 14(24):11551–11565, 2006.

[10] K. Sayood et al. Statistical evaluation of image quality measures. *Journal of Electronic imaging*, 11(2):206–223, 2002.

[11] Y. Sutcu, S. Bayram, H. T. Sencar, and N. Memon. Improvements on sensor noise based source camera identification. In *Multimedia and Expo, 2007 IEEE International Conference on*, pages 24–27. IEEE, 2007.

[12] L. T. Van, S. Emmanuel, and M. S. Kankanhalli. Identifying source cell phone using chromatic aberration. In *Multimedia and Expo, 2007 IEEE International Conference on*, pages 883–886. IEEE, 2007.

PRILOGE

A. GENERIRANJE ZNAčILK

Spodnja izvorna koda je bila uporabljene za generiranje lastnosti iz izbranega nabora fotografij.

```
from skimage import io
import numpy as np
from os import listdir
import fnmatch

# path to directory containing images
dir = "Dropbox/photos_dataset"

# get all JPG files from selected directory
photos = fnmatch.filter(listdir(dir), '*.JPG')
print("Processing", len(photos), "images...")

#photos = photos[:10]

# extract model names from file names
photos_model_names = ["".join(p.split("_")[:2])
        for p in photos]

rgb_apv = []
rg_cor = []
gb_cor = []
rb_cor = []

for photo in photos:

    # load selected image
    image = io.imread(dir + "/" + photo)

    # for simplicity, reshape 2D array of
    # pixels into 1D array
    reshaped_image = image.reshape(
      image.shape[0]*image.shape[1],
        image.shape[2])

    # rgb_apv - average pixel value
    # for each RGB channel
    rgb_apv.append(np.mean(reshaped_image,
      axis = 0))

    # RGB pairs correlations
    rg_cor.append(np.correlate(
      reshaped_image[:, 0],
        reshaped_image[:, 1]))
    gb_cor.append(np.correlate(
      reshaped_image[:, 1],
      reshaped_image[:, 2]))
    rb_cor.append(np.correlate(
      reshaped_image[:, 0],
        reshaped_image[:, 2]))

# select features to be written and save them
features_with_class = np.column_stack((
   rgb_apv,
   rg_cor,
      gb_cor,
      rb_cor,
      photos_model_names))
np.savetxt("features.csv",
   features_with_class,
        fmt='%s', delimiter=",")
```

Detekcija modificiranih slik s tehniko lokalnih binarnih vzorcev

Matej Kopar
Fakulteta za računalništvo in informatiko
Večna pot 113
Ljubljana, Slovenija
mk0178@student.uni-lj.si

Anže Medved
Fakulteta za računalništvo in informatiko
Večna pot 113
Ljubljana, Slovenija
am1947@student.uni-lj.si

Dejan Štepec
Fakulteta za računalništvo in informatiko
Večna pot 113
Ljubljana, Slovenija
ds2268@student.uni-lj.si

POVZETEK

Metode, kot je rezanje šivov, ki je zelo priljubljena za spreminjanje velikosti slik, tako da ohranjamo bistvo slike, so dandanes zelo priljubljene in jih najdemo praktično v vsakem programu za obdelavo slik. Te metode pa so lahko tudi zlorabljene, tako, da se z njimi spremeni pomen slik. V delu predstavimo različne načine detekcije sprememb v slikah, ki so posledica rezanja šivov. Obstaja veliko različnih načinov detekcije. V našem delu predstavimo tri načine, ki so se v zadnjem času izkazale kot najuspešnejše.

Najprej predstavimo kaj je rezanje šivov, ter kako lahko algoritem uporabimo za namerno spreminjanje vsebine slik. V nadaljevanju implementiramo pristop z binarnimi lokalnimi vzorci in energijskimi značilkami, ki predstavljajo vhod v učni algoritem, s katerim ugotovimo, ali je bila slika modificirana z metodo rezanja šivov. Metodo testiramo pri različnih testnih pogojih.

Uporabili smo protokol testiranja, kot ga uporabljajo v vseh člankih s tega področja, kar omogoča primerjavo med različnimi metodami. Protokol testiranja smo tudi izboljšali, tako da smo testiranja izvedli na način, ki bolje izražajo realno uporabo metod detekcije, saj smo ugotovili, da je protokol, ki ga uporabljajo raziskovalci precej generičen in ne predstavlja realne ocene metod. Poleg tega smo raziskavo naredili na način, ki omogoča enostavno ponovljivost rezultatov.

Keywords

digitalna forenzika, rezanje šivov, detekcija modifikacij, BLV, SVM

1. UVOD

Forenzika digitalnih slik je področje, ki poskuša slepo detektirati nedovoljene posege v slikah. Vstavljanje slik v druge slike je eden najboljših pristopov za spremembo sporočila, ki ga vsebuje slikovni medij. Vse več programske opreme za urejanje slik omogoča enostavno ustvarjanje sestavljenih slik z rezultati, ki so komaj opazni človeškemu očesu. Tehnike prepletanja so uporabne za maskiranje ločnic dodanih regij. Ena od možnosti kreacij sestavljenih slik je tudi geometrijska transformacija. Ko kreiramo novo sliko, je le-to potrebno večkrat ponovno vzorčiti (spremeniti velikost, jo rotirati ali skrčiti), da izgleda naravno oz. da lahko zagotovimo, da dodan objekt upošteva perspektivo prvotne slike. Ponovno vzorčenje proizvede artefakte v slikovnih histogramih, kar zagotavlja uporabno iztočnico za detekcijo sestavljenih slik.

Upoštevati moramo tudi, da vnešeno gradivo ne prihaja nujno iz naravnih slik. Ker se računalniška grafika vse bolj razvija, lahko ustvarimo vse več realističnih 3D objektov, ki bi jih lahko vstavili v sestavljene slike.

Vse več raziskovalcev se ukvarja s tehniko spreminjanje velikosti slike glede na vsebino (angl. Content aware image resizing oz. CAIR). Med CAIR tehnikami je tehnika rezanja šivov ena od najbolj široko uporabljenih.
Ob spreminjanju velikosti slike ostane v tehnikah spreminjanja velikosti z zavedanjem vsebine pomembna vsebina nespremenjena. Predvideno je, da pomembna vsebina ni predstavljena s piksli z nizko energijo.
Pri metodi je šiv definiran kot povezana pot pikslov z enim pikslom na vrstico (stolpec) in prehajanjem slike od zgoraj navzdol (oz. od leve proti desni). Vsako brisanje šiva sovpada z vodoravno oz. navpično spremembo velikosti za en piksel, kar vodi v končni rezultat bolj smiselnih slik kot bi jih dobili zgolj s ponovnim vzorčenjem. Za spremembo velikosti predstavimo vrsto optimalnih 8-povezanih poti pikslov, katere prehajajo celotno sliko navpično ali vodoravno. Optimalni kriterij za odstranitev šiva je povezan z energijsko funkcijo, izračunano za vse točke okrog šiva. Optimalni šiv ne spremeni kvalitete slike med spreminjanjem velikosti. Slike, kjer je bila vsebina le-te oz. njena dimenzija spremenjena, obravnavamo kot sliko z nedovoljenim posegom.
Pri uporabi algoritma na določeni regiji oz. objektu, ta metodologija postane izredno natančno orodje za odstranjevanje objektov. To dosežemo z iterativnim brisanjem vseh šivov ob prehodu čez določeno območje. Zaradi uspešnosti tehnike je bilo rezanje šivov integrirano v veliko komericalnih programov za obdelavo slik, npr. Adobe Photoshop, GIMP, ImageMagic in iResizer. Če je odstranjen objekt v splošnem semantični objekt, se lahko kontekst slike popolnoma spremeni (lahko se spremeni semantična vsebina, ki jo slika sporoča).
Semantika slike je lahko spremenjena tudi z apliciranjem

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

enostavnih tehnik, kot so manipulacija histogramov ali okrepitev kontrasta. V splošnem velja, da apliciranje filtrov ali enostavnih geometričnih transformacij lahko ogrozi forenzično analizo s prekrivanjem ali brisanjem sledi z nadalnjim posegom v slike.
Rezanje šivov je lahko uporabljeno tudi za odstranjevanje objektov iz slik v nedovoljene namene. Zaradi tega je zelo pomembna izdelava metode, ki zaznava spremembo slike s tehniko rezanja šivov. Omeniti je potrebno, da v primeru detektiranja nedovoljenih posegov ne obstaja merilo uspešnosti. Raziskovalci iz Columbia University so v letu 2004 ustvarili Podatkovni nabor prvotnih in obrezanih slikovnih blokov (Dataset of authentic and spliced image blocks) s 933 avtentičnimi in 912 obrezanimi bloki slike v velikosti 128x128 pikslov, pridobljenih iz nabora slik CalPhotos. Ta podatkovni nabor je uporaben za raziskovalce, ki se ukvarjajo z detekcijo sestavljenih slik. V poglavju 3 opišemo metode modifikacij šivov, tj. rezanje in dodajanje šivov, v poglavju 4 opišemo metode za detekcijo modifikacij. V poglavju 5 opišemo našo implementacijo sistema za detekcijo in v poglavju 6 rezultate naše implementacije.

2. SORODNA DELA

V delu [9] je raziskano, kako ponovno vzorčenje (povečanje, zmanjšanje, rotiranje ipd.) vpliva na specifične statistične korelacije. V spremenjeni sliki niso nujno prisotni s prostim očesom vidni artefakti. Kljub temu pa vsaka slika vsebuje nekatere specifične statistike, ki veljajo v naravnih in nespremenjenih slikah. Z metodo, predlagano v delu [9] je pokazano, kako so korelacije lahko avtomatsko detektirane z algoritmom z maksimizacijo pričakovane vrednosti (angl. Expectation Maximization algorithm).
V [4] je predstavljena detekcija s pridobivanjem značilk na osnovi energijske pristranskosti. Nabor testiranih slik v tem delu je vseboval slike, ki so bile spremenjene z namenom, da bi zakrili informacije o kaznivih dejanjih. Običajno so elementi, ki nakazujejo kaznivo dejanje na slikah eni od pomembnejših elementov zaradi višje sporočilne vrednosti in imajo zato tudi višje energijske vrednosti. Algoritem s pomočjo energijske pristranskosti v slikah detektira te spremembe. S predlagano metodo je dosežena detekcijska natančnost med 84,0% in 91,3% za slike s spremenjeno velikostjo in skalirnim razmerjem med 20% in 30%.
V delu [3] je predlagana še ena detekcijska metoda za detektiranje rezanja šivov v JPEG slikah - metoda blokirajočih artefaktnih karakteristik matrike (angl. blocking artifact characteristics matrix - BACM). Vsak element na sliki, še posebej pa tisti, ki vsebujejo semantični pomen slike, morajo biti glede na merilo oz. velikost ostalih objektov smiselno simetrični. Metoda, predstavljena v delu [3] pa temelji na predpostavki, da lahko z rezanjem šivov v sliki uničimo to naravno simetrično lastnost pomembnih elementov v slikah in tako ustvarimo dodatne artefakte, ki jih algoritem zazna.
V delu [5] je raziskana aktivna forenzična metoda za detekcijo slik, spremenjenih s tehniko rezanja šivov. Uporabili so tehniko forenzične zgoščevalne funkcije, ki uporabi stransko informacijo slike. V primerjavi z drugimi multimedijskimi forenzičnimi pristopi je potrebno tukaj varno pripeti forenzično zgoščevalno vrednost k sliki. Eksperimentalno so odkrili, da lahko s forenzično zgoščevalno funkcijo ocenimo količino izrezanih šivov ter njihove približne lokacije. S pridobljenimi rezultati so uspeli rekonstruirati prvotne slike iz izrezanih slik. Kljub temu pa ima predlagan pristop dve slabi lastnosti - ponarejevalci lahko brez težav odstranijo forenzično zgoščevalno funkcijo če je dodana glavi slikovne datoteke in ker je forenzična zgoščevalna funkcija aktivna metoda, mora biti vgrajena vnaprej. Aktivna forenzična metoda mora biti namreč ustvarjena ob času posnetka oz. ob času pošiljanja in poslana poleg slike.
Zgoraj opisani pristopi ne zadovoljijo popolnoma potreb današnje forenzike. Če se upošteva vse pomembne lastnosti rezanja šivov, se lahko še vedno izboljša detekcijsko natančnost. Nekatere tehnike nedovoljenega spreminjanja slik ne vnesejo nujno tudi vizualno motečih artefaktov, kot so motne sence in zamegljenost, kar pa je glavni izziv za slepo detekcijo slik z rezanjem šivov. V večini primerov odstranjevanje šivov iz slike zgolj spremeni lokalne teksture in energijsko porazdeljenost, vendar je ključna težava pri detekciji rezanja šivov iskanje občutljivih značilk, ki izrazijo neločljivo naravo tehnike rezanja šivov. Zato so v delu [13] predlagali nov pristop detekcije slik, spremenjenih s tehniko rezanja šivov. Implementirali so metodo z uporabo binarnih lokalnih vzorcev oz. BLV (angl. local binary patterns). S predlagano metodo so dosegli bistveno boljše rezultate kot v predhodno omenjenih metodah, zato smo se odločili za implementacijo po tem članku, namesto za implementacijo po članku [12], ki je bil izvorno predlagan s strani mentorjev.

3. MODIFIKACIJA ŠIVOV

Pri naivnem spreminjanju velikosti slike pride ponavadi do popačitve objektov, bodisi zaradi spremembe razmerja v stranicah, bodisi zaradi prevelike stopnje skaliranja slike. Eden izmed pristopov, pri katerem se objekti na sliki ne popačijo je obrezovanje slike, vendar smo omejeni le na odstranjevanje pikslov na robu slike. Omejitvi pa se lahko izognemo z uporabo rezanja šivov, kjer poleg geometrijskih lastnostih upoštevamo tudi vsebino slike.

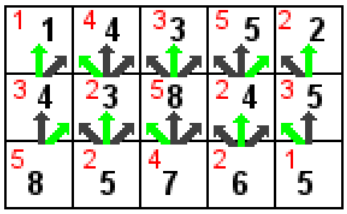
Glavna ideja rezanja šivov temelji na izbrisu poti pikslov, od zgornjega do spodnjega robu oz. od levega do desnega robu slike, ki nosi najmanj informacije. Količina informacije, ki jo piksli nosijo, je izračunana na podlagi energijske funkcije. Algoritem je podrobneje opisan v podpoglavju 3.1, obratni algoritem, torej dodajanje šivov, pa je opisan v podpoglavju 3.2.

3.1 Rezanje šivov

Kadar želimo sliko preoblikovati tako, da bo nosila enako semantično informacijo kot pred preoblikovanjem, se postavi vprašanje kako določiti pomembnost posameznih pikslov. Znano je, da smo ljudje bolj občutljivi na spremembe v intenziteti kot v barvi, zato se bodo pomembnejši piksli nahajali v nehomogenih regijah. Za določanje homogenosti regije lahko uporabimo gradient, in sicer v vertikalni in horizontalni smeri, saj bi radi v teh dveh smereh odstranjevali piksle. Funkcijo pomembnosti torej predstavlja vrednost gradienta.

Sedaj, ko vemo, kateri piksli so manj pomembni, se postavi vprašanje kako jih odstraniti. Z upoštevanje intuicije, bi se verjetno odločili za zaporedno odstranjevanje pikslov z najmanjšo vrednostjo energijske funkcije. Izkaže se, da tovrsten pristop vodi k popačenju, kot ga dobimo pri naivnem skaliranju slike. Z namenom ohranjanja konsistentnosti slike in preprečevanju popačenja uporabimo slikovne šive.

Algoritem za detekcijo šivov je tako sestavljen iz dveh korakov. V prvem koraku izračunamo kumulativno energijsko vrednost za vsak piksel, pri čemer z izračunom začnemo na



Slika 1: Izračun kumulativne energije pikslov.

zgornjem levem kotu slike. Kumulativna energijska vrednost piksla izračunamo kot vsoto energije piksla in minimalne vrednosti energije predhodnika. Postopek izračuna je prikazan na sliki 1, kjer puščice označujejo predhodnike, in sicer zelena puščica predstavlja izbranega predhodnika. V rdeči barvi so zapisane energije posameznih pikslov, v črni pa njihove kumulativne vrednosti. V drugem koraku algoritma začnemo na spodnjem robu slike in izberemo piksel z najnižjo kumulativno vrednostjo, nato pa se premikamo proti vrhu slike preko predhodnikov z najnižjo vrednostjo. Pot, ki smo jo našli s pomočjo algoritma, imenujemo šiv in je definiran kot povezana pot pikslov, od spodnjega do zgornjega robu slike, ki imajo najnižjo energijo. Najden šiv lahko nato odstranimo. Enak postopek lahko ponovimo tudi v horizontalni smeri, tako da sliko enostavno transponiramo.

Drugi korak algoritma ponavljamo dokler ne odstranimo želenega števila šivov. Celoten postopek izbrisa šivov je prikazan tudi na sliki 2. Prvi del slike prikazuje izvorno sliko, drugi del energijske vrednosti posameznih pikslov, tretji del izbrane šive in zadnji del modificirano sliko.

3.2 Dodajanje šivov

Algoritem za dodajanje šivov je skoraj identičen algoritmu brisanja šivov, razlikujeta se le v drugem koraku. Če želimo v sliko dodati k šivov, torej jo povečati za k pikslov v posamezni vrstici oz. stolpcu, moramo vse šive najprej poiskati, nato pa izvesti postopek dodajanja. Šiv dodamo tako, da vrednosti pikslov na šivu dupliciramo in pomaknemo preostale piksle v desno.

V primeru, da bi šive dodajali sekvenčno, enega za drugim, bi se zgodilo, da bi dodajali piksle vedno na isti lokaciji v sliki, saj bi vedno našli isto pot z minimalno energijsko vrednostjo.

4. DETEKCIJA MODIFIKACIJ

4.1 Forenzična razprševalna funkcija

Raziskovanje forenzične razprševalne funkcije (angl. Forensics hash) je povezano z robustnim slikovnim razprševanjem. Običajno se uporablja pri skriti multimedijski forenziki (angl. blind forensics). Skrite forenzične tehnike odgovorijo na forenzična vprašanja brez podanih eksplicitnih oznak, vendar običajno zahtevajo večjo računalniško zmogljivost. Tradicionalno slikovno razprševanje uporablja krajše oznake oz. podpise za uspešno avtentikacijo slike. Dobra arhitektura forenzične razprševalne funkcije prebere t.i. stranske informacije iz slik. S to metodo dobimo boljše forenzične rezultate, prav tako pa je metoda učinkovitejša od metod skrite forenzike.

Za modularnost in razširljivost lahko forenzična razprševalna funkcija vsebuje več komponent – vsaka od komponent lahko opravlja specifično operacijo, ki je lahko drugi operaciji komplementarna ali pa deluje z njo v sinergiji. Za konstrukcijo forenzične zgoščevalne funkcije se lahko uporabi pristop s kompaktno predstavitvijo stabilnih SIFT značilk v sliki. Značilke SIFT z visokimi vrednostmi kontrastov so robustne za večino slikovnih operacij, npr. karakteristična skala in dominantna orientacija SIFT točk se lahko uporabi za oceno geometrične transformacije (rotacija ali skaliranje).

Forenzična zgoščevalna funkcija se generira v več korakih, ki so natančneje opisani v delu [5]. Prednost njihovega pristopa, pred sorodnimi pristopi, je višja natančnost in geometrijska robustnost transformacije, ki omogoči nadaljno forenzično analizo.

4.2 BACM analiza

Metoda BACM se prav tako kot večina ostalih metod, opisanih v tem delu, uporablja za detekcijo spremenjenih slik oz. odstranjevanju objektov v slikah. BACM je blokirajoči artefakt karakteristične matrike (angl. Blocking Artifact Characteristics Matrix) ki vsebuje simetrično lastnost blokirajočih artefaktov v JPEG slikovnem formatu [6]. Običajno se metoda uporabi za iskanje vektorjev značilk pri detekciji slik, ki so bile spremenjene z metodo rezanja šivov. Če je bil šiv odstranjen je lahko kontinuiteta slike uničena, še posebej če gre za blokirajoč efekt.

V delu [12] je bila predstavljena raziskava z uporabo BACM metode. Avtorji predstavijo razvito arhitekturo, ki so jo uporabili za detekcijo spremenjenih slik. V prvem delu članka opišejo teoretično ozadje blokirajočih efektov v JPEG slikah, sledi primer uporabe BACM za merjenje simetričnih lastnosti blokirajočih artefaktov, predstavijo pa tudi dvaindvajset značilk s katerimi opišejo posamezno sliko. Za klasifikacijo v posamezen razred uporabijo klasifikator SVM.

4.2.1 Arhitektura sistema

V tem poglavju je predstavljena arhitetkura sistema, ki je bil razvit v delu [12].

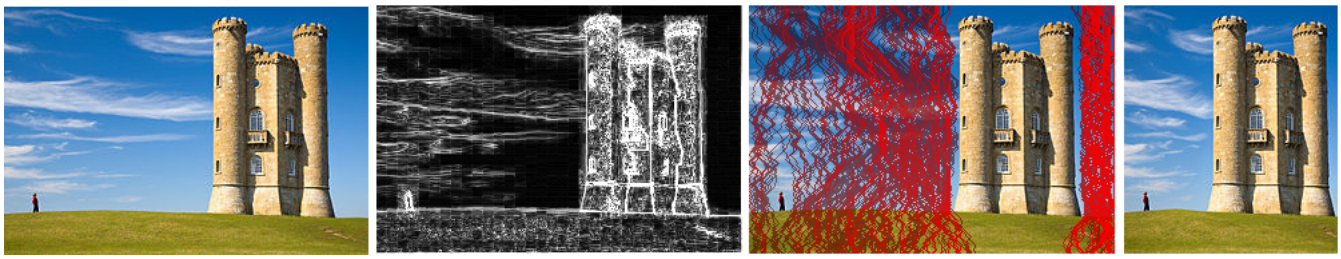
1. Podatkovni nabor slik
   Začetne slike so dobili iz baze UCID [11] in baze UCUS. Slike iz baze UCID so v formatu .tif (nekompresirane slike), medtem ko so slike iz baze UCUS v .jpg formatu (kompresirane slike) z neznanimi ali nedefiniranimi faktorji kvalitete. Vse slike nato pretvorijo v format JPEG.

2. Slikovni prostor
   Vse slike so pretvorili v slikovni prostor YCbCr. Po podvzorčenju za kompresijski format JPEG so ugotovili, da svetilnost, ki je predstavljena v komponenti Y, ni bila spremenjena, zato so analizo usmerili predvsem v to komponento.

3. Statistični izračun vrednost blokirajočih efektov
   Glavna ideja tega koraka je izračunati različne vrednosti (npr. statistiko porazdelitve intenzitete) za posamezen blok.

4. Izračun BACM
   V delu so izračunali matriko, da bi pokazali simetrični fenomen blokirajočih efektov. Cilj tega koraka je bil izračun dvaindvajsetih vektorjev značilk za identifikacijo simetrije matrike.

5. SVM učenje
   Dvaindvajset značilk so naučili in skonstruirali klasifi-



Slika 2: Posamezne faze izbrisa šivov.

kacijski model za identifikacijo slik v podatkovne naboru.

6. Identifikacija
   V zadnjem koraku so naučeni model uporabili za testiranje testnega nabora z uporabo prečnega preverjanja.

4.3 Pristop z binarnimi lokalnimi vzorci

Binarni lokalni vzorci oz. BLV (angl. local binary patterns) so lokalni deskriptorji, ki so širše uporabni v aplikacijah za slikovno analizo [8, 7]. BLV je enostaven, ampak zelo učinkovit teksturni operator, ki z upragovanjem sosednih pikslov označi piksle slik in predstavi rezultat kot binarno število. Zaradi računske enostavnosti je BLV teksturni operator postal priljubljen pristop v več aplikacijah. Združimo ga lahko s tradicionalnimi divergentnimi statističnimi in strukturnimi modeli teksturnih analiz. Najpomembnejša lastnost BLV je verjetno robustnost za monotone sivinske spremembe. Prav tako je pomembna lastnost računska enostavnost, ki omogoča analizo slik v zahtevnih nastavitvah v realnem času. BLV je izredno zmogljivo orodje za teksturne klasifikacije - v kombinaciji z deskriptorjem histograma orientiranih gradientov (angl. Histogram of oriented gradients) izboljšuje detekcijsko zmogljivost na določenih podatkovnih naborih.

Splošen koncept BLV vektorja je naslednji:

1. Okno razdelimo v posamezne celice
2. Znotraj celice primerjamo vrednost centralnega piksla z ostalimi piksli
3. Kjer je vrednost centralnega piksla večja od vrednosti njegovih sosedov, zapišemo 0, drugače zapišemo 1. Števila preberemo v smeri urinega kazalca in tako dobimo 8-bitno binarno število.
4. Izračunamo histogram čez celico s frekvenco pojavitve vsake številke.
5. Združimo histograme vseh celic - dobimo vektor značilk za celotno okno.

Vektor značilk lahko sedaj obdelujemo s pomočjo metode SVM oz. katerega drugega algoritma za klasifikacijo. Podrobnejši opis pristopa z BLV in konkretna implementacija je opisana v poglavju 5.

5. IMPLEMENTACIJA

Implementirali smo pristop z binarnimi lokalnimi vzorci (BLV), saj kot je omenjeno v prejšnjem poglavju, se pri rezanju šivov spremeni lokalna tekstura v sliki. Z BLV predprocesiramo vhodno sliko in tako izrazimo lokalne značilnosti slike. Nato izračunamo različne energijske značilnice, ki jih podrobneje predstavimo v naslednjem poglavju. Poleg 18 standardnih značilnic, ki jih uporablja večina avtorjev, smo dodali tudi 6 značilnic, izpeljanih na tako imenovanih pol-slikah, ki so bile predstavljene v [13]. Ker so značilnice izračunane v domeni BLV, izražajo tako lokalne značilnosti kot globalno distribucijo energije.

V nadaljevanju smo te značilnice uporabili pri učenju klasifikatorja SVM, ki je testne slike klasificiral glede na vhodne učne podatke na slike, ki so bile modificirane z brisanjem ali dodajanjem šivov in pa slike, ki niso bile modificirane. Celoten proces je grafično predstavljen na sliki 3.

5.1 Značilke

Pri rezanju šivov odstranjujemo šive z najnižjimi energijskimi vrednostmi, torej spreminjamo distribucijo energije, saj ima taka sliko višjo energijsko vrednost, kot slika, ki ni bila modificirana. Posledica odstranjevanja šivov je tudi, da je porazdelitev bistveno bolj enakomerna. Značilke so torej enostavne statistične vrednosti ki opisujejo te značilnosti, s katerimi ločujemo med modificirano in originalno sliko. V nadaljevanju predstavimo nekaj značilnic in njihove lastnosti.

V [10] je predstavljenih prvih 18 statističnih značilnic, ki smo jih uporabili. Razdelimo jih lahko v tri skupine, glede na domeno računanja: energijske značilnice, značilnice šivov in značilnice šuma. Energijske značilnice so npr. povprečna energija slike, povrečna energija stolpcev in vrstic slike, povprečna razlika energij. Značilnice šivov računamo po vrsticah in stolpcih, tako da izračunamo komulativno energijsko matriko šivov, za vse možne šive, na tej matriki nato poračunamo pet statističnih vrednosti in sicer: minimum, maksimum, povprečna vrednost, standardni odklon in razlika med maksimumum in minimumom tako po stolpcih kot vrsticah matrike. Tako dobimo 10 značilnic šivov. Značilnice šuma temeljijo na tem, da z odstranjevanjem šivov spremenimo količino šuma v sliki. Značilnice poračunamo na podoben način kot značilnice šivov le da vhodno matriko predstavlja slika, ki jo dobimo kot razliko med sliko, ki jo želimo klasificirati in isto sliko, ki jo filtriramo z Winner filtrom (okno 5 x 5), več podrobnosti o značilkah lahko najdete v originalnem delu [10]. Poleg članka je objavljena tudi programska koda za računanje značilnic, ki smo jo uporabili.

Dodali smo še 6 značilnic, ki so predstavljene v [13]. Značilnice so izračunane na podoben način, le da namesto celotnih šivov uporabijo zgolj polovico. Razlog za to je, da pot z najnižjo energijo na celotni sliki ni enaka poti polovice slike . Torej ko odstranjujemo šiv na celotni sliki, ne odstranjujemo optimalnega šiva na npr. zgornji polovici slike. Tako izračunane značilnice šivov do polovice slike bolje opišejo lokalne artefakte in dvignejo natančnost detekcije.

Slika 3: Proces detekcije modificiranih slik, kot smo ga implementirali.

Slika 4: Primer modificirane slike, odstranjevanje semaforja.

5.2 Izbris objektov

Poleg klasične uporabe rezanja ali dodajanja šivov za za tako imenovano vsebinsko zmanjševanje ali povečevanje slike, se lahko rezanje šivov uporablja tudi za odstranjevanje objektov. Ta postopek se velikokrat uporablja pri ponarejanju slik, tako, da se spremeni semantični pomen slike. Obstaja več različnih načinov implementacije, v članku [13] so uporabili enostaven postopek, tako da so izračunali vse poti šivov, ter nato odstranili vse šive, ki potekajo skozi označen objekt, ki so ga označili za brisanje. Drugi način je, da območju, ki ga želimo izbrisati zelo zmanjšamo energijsko vrednost, tako, da bodo šivi potekali skozi območje, ki ga želimo izbrisati. Za bolj natančno izrezovanje objektov lahko uporabljamo tudi dvigovanje energijske vrednosti, za ohranitev določenih delov objekta. Za ta namen smo uporabili program Seam Carving GUI [2]. Primer odstranjevanja objekta predstavlja slika 4.

5.3 Klasifikacija

Klasifikacijo na podlagi pridobljenih značilk smo izvedli z metodo podpornih vektorjev (SVM - angl. support vector machine). Izhodišče za nastanek SVM je množica učnih primerov, za katere je znano kateremu razredu pripadajo. Vsak učni primer predstavimo z vektorjem v vektorskem prostoru, kjer so komponente vektorja predstavljene z vrednostmi posameznih značilk. Dimenzija vektorskega prostora tako ustreza številu značilk, ki pripadajo primeru. Naloga SVM je poiskati v n dimenzionalnem prostoru hiperravnino , ki ločuje primere iz različnih razredov. Razdalje vektorjev, ki ležijo najbližje hiperravnini, pri tem maksimiramo. Pri postavljanju hiperravnine ponavadi ne upoštevamo vseh učnih primerov, saj vektorji, ki so daleč od hiperravnine oz. so skriti za fronto ostalih ne vplivajo na njeno lego. Torej je lega hiperravnine odvisna le od njej najbližjih vektorjev, ki jim rečemo podporni vektorji.

Za implementacijo metode podpornih vektorjev smo uporabili knjižnico LIBSVM [1], ki je ena izmed najbolj uporabljenih knjižnic na tem področju in podpira tudi vmesnik za Matlab.

6. REZULTATI KLASIFIKACIJE

V pričujočem poglavju najprej opišemo podatke, ki smo jih uporabili za testiranje, nato pa predstavimo testne scenarije in analiziramo pridobljene rezultate.

6.1 Priprava podatkov

Ker po naših podatkih trenutno ni na voljo prosto dostopne podatkovne baze slik, ki bi se uporabljala za namene evalvacije forenzičnih pristopov, smo uporabili slikovne bazo, ki je bila predstavljena in uporabljena v člankih - UCID (angl. Uncompressed image database) [11]. V slikovni bazi se nahaja 1338 ne stisnjenih slik, ki imajo ločljivost 384 x 512 pikslov. Fotografije so raznolike in predstavljajo pokrajine, živali, ljudi, stavbe, ...

Takoj po prenosu smo slike razdelili na testno in učno množico. Za učno množico smo izbrali prvih 1000 slik, za testno pa zadnjih 338, tako da med množicama nikoli ni prišlo do prekrivanja. Za namene validacijske množice smo sami dodali še 50 naključnih slik. V učni množici smo 600 slik modificirali z rezanjem šivov, ostalih 400 pa smo pustili nespremenjenih, tako da se je klasifikator učil na pozitivnih in negativnih primerih. Podobno smo storili tudi pri testni množici, kjer smo 180 slik modificirali, 153 slik pa obdržali za referenco.

Testiranje smo razdelili na tri scenarije. V prvem scenariju smo na učnih in testnih primerih izvedli enako stopnjo skaliranja z izbrisom šivov, v drugem scenariju smo testirali modele naučene na različnih stopnjah skaliranja z naključno skaliranimi slikami in v zadnjem sklopu testirali model, ki se je učil na vseh stopnjah skaliranja, z naključno skalirano testno množico. Zadnji scenarij je po našem mnenju najbolj realen, saj v realnem okolju ne moremo v naprej pričakovati stopnje skaliranja.

V sledečih podpoglavjih so predstavljeni posamezni sklopi

testiranja in analiza pridobljenih rezultatov.

6.2 Klasifikacija z enako stopnjo skaliranja v učni in testni množici

V prvem scenariju smo slike v učni in testni množici skalirani z enako stopnjo izbrisa šivov, ter tako naredili analizo uspešnosti modela v idealnih primerih, ki pa seveda niso preveč realni. Ker so naše značilke lokalno občutljive, smo se odločili da analizo ločimo na dva dela. V prvem so obravnavali nizke stopnje izbrisa, torej od 1% do 13%. V drugem sklopu pa obravnavamo višje stopnje izbrisa šivov, torej od 16% do 30%.

6.2.1 Nizke stopnje izbrisa

V razpredelnici 1 so prikazani rezultati, ki smo jih dobili pri nizkih stopnjah izbrisa. Pridobljeni rezultati so ekvivalentni tistim, ki so predstavljeni v referenčnih člankih. Točnosti napovedi so pri nas nižje za faktorje skaliranja od 10% dalje, vendar sumimo, da je to posledica načina izvedbe eksperimentov. V izvornih člankih so poleg skaliranih slik podali modelu še iste originalne slike, torej klasifikacijski model je imel vsako sliko podano dvakrat - modificirano in ne-modificirano. V našem eksperimentu modelu nikoli nismo podali slike dvakrat, saj smo za referenčne, torej nemodificirane slike, podali druge slike, kot smo to opisali v poglavju 6.1.

V razpredelnici smo poleg točnosti predstavili tudi delež napačno napovedanih negativnih primerov (FNR - angl. false negative rate), delež napačno napovedanih pozitivnih primerov (FPR - angl. false positive rate), delež pravilno napovedanih pozitivnih primerov (TPR - angl. true positive rate) in delež pravilno napovedanih negativnih primerov (TNR - angl. ture negative rate). Kot pozitiven primer smo privzeli primer, ko je slika modificirana. Za skalirna faktorja 1% in 4% vidimo da je TNR enak 1, kar pomeni, da je vsak primer, ki je bil klasificiran kot negativen (torej slika ni bila modificirana), klasificiran pravilno, kljub le približno 54% točnosti. Podatki so predstavljeni tudi v obliki ROC krivulje na sliki 5.

ROC krivulje prikazujejo pragove za deleže pravilno napovedanih pozitivnih primerov (TP) in napačno pozitivnih primerov (FP), ki so določeni glede na delež pravilno napovedanih skaliranih slik in delež nepravilno napovedanih nemodificiranih slik. Idealna točka ROC krivulje bi se nahajala pri vrednostih $TPR = 1$ in $FPR = 0$.

Tabela 1: Rezultati pri enakih stopnjah skaliranja.

Skalirni faktor	Točnost	FNR	FPR	TPR	TNR
1%	0.539	0	0.464	0.536	1
4%	0.556	0	0.455	0.545	1
7%	0.565	0.133	0.449	0.551	0.867
10%	0.624	0.228	0.406	0.594	0.772
13%	0.672	0.209	0.366	0.634	0.790

6.2.2 Visoke stopnje izbrisa

V razpredelnici 2 so predstavljeni rezultati pridobljeni pri visokih stopnjah izbrisa. Točnosti pri teh primerih so višje kot pri nižjih stopnjah, kar je bil pričakovan rezultat, saj pride do večje spremembe lokalne teksture. Pri stopnjah iz razpredelnice 2 so mnogokrat vidni tudi artefakti na slikah, ki so nastali ob izbrisu šivov, tako da bi take slike brez težav klasificiral tudi brez uporabe modela.

Najvišjo točnost dosežemo pri 30% skaliranju, kjer znaša skoraj 90%. Vidimo tudi, da trend visoke vrednosti TNR ostaja tudi v tem sklopu testiranja, tako da lahko za sliko, ki je bila klasificirana kot negativna, z visokim zaupanjem trdimo, da ni bila spremenjena. Podatki so predstavljeni tudi v obliki ROC krivulje na sliki 5.

Tudi v tem primeru smo pridobili ekvivalentne, vendar nižje stopnje točnosti, kot so bile navedene v izvornih člankih.

Tabela 2: Rezultati pri enakih stopnjah skaliranja.

Skalirni faktor	Točnost	FNR	FPR	TPR	TNR
16%	0.713	0.157	0.333	0.667	0.843
20%	0.790	0.061	0.272	0.728	0.939
23%	0.834	0.045	0.226	0.774	0.955
26%	0.861	0.041	0.194	0.806	0.959
30%	0.899	0.030	0.146	0.854	0.970

6.3 Klasifikacija z mešanim skaliranjem testne množice

Z namenom testiranja univerzalnosti in robustnosti pristopa, smo v drugem scenariju testirali modele naučene na različnih stopnjah skaliranja s testnimi množicami, ki vsebujejo mešane skalirne faktorje. Skalirni faktorji v testni množici so na intervalu med 1% in 30%. Tudi v tem primeru smo se odločili da analizo ločimo na dva dela, tako kot v predhodnem podpoglavju.

Pridobljeni rezultati so prikazani v razpredelnici 3 in 4, ter sliki 6.

Tabela 3: Rezultati pri mešanih stopnjah skaliranja.

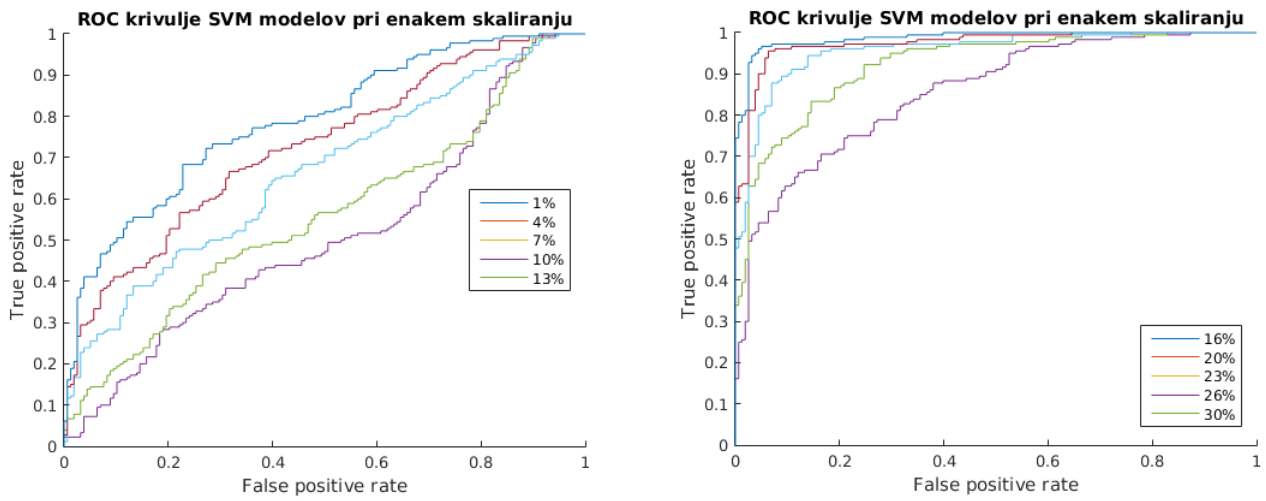
Skalirni faktor	Točnost	FNR	FPR	TPR	TNR
1%	0.538	0	0.464	0.536	1
4%	0.556	0	0.455	0.545	1
7%	0.568	0.071	0.448	0.552	0.929
10%	0.630	0.200	0.403	0.597	0.800
13%	0.660	0.247	0.371	0.629	0.753

Tabela 4: Rezultati pri mešanih stopnjah skaliranja.

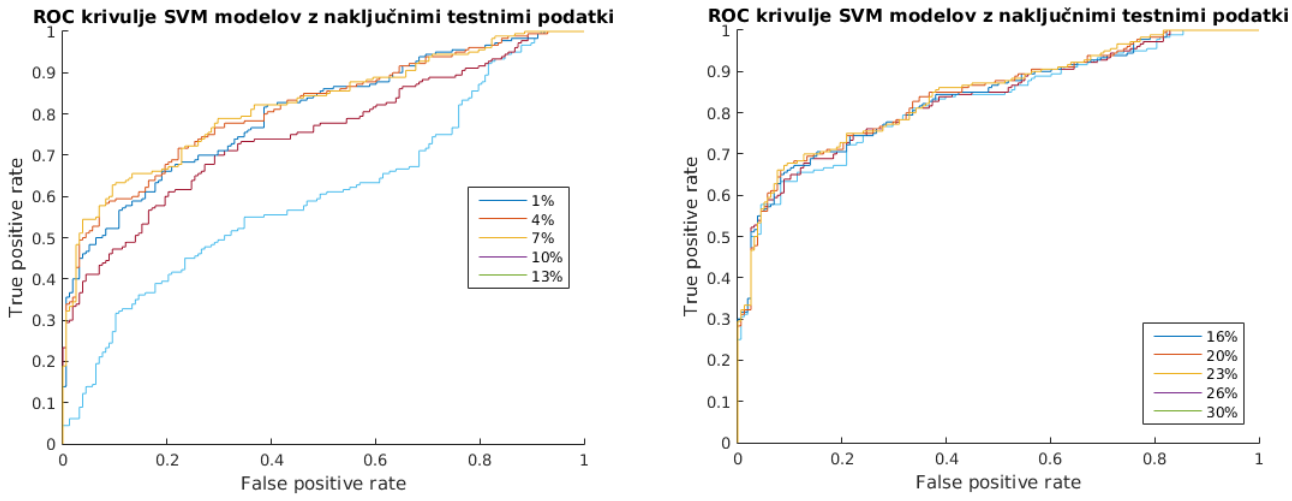
Skalirni faktor	Točnost	FNR	FPR	TPR	TNR
16%	0.675	0.265	0.352	0.648	0.735
20%	0.725	0.231	0.299	0.701	0.769
23%	0.734	0.267	0.266	0.734	0.733
26%	0.749	0.270	0.235	0.765	0.730
30%	0.757	0.289	0.190	0.810	0.711

Iz razpredelnic lahko razberemo, da so klasifikacijske točnosti skoraj enake pri nižjih koeficientih skaliranja, medtem ko so pri višjih koeficientih nižje. Takšen primer, je veliko bolj realen kot predhodni, kjer smo imeli enak koeficient rezanja šivov v učni in testni množici. Opazimo tudi, da z naraščanjem velikosti koeficienta rezanja pada vrednost pravilno napovedanih negativnih primerov in narašča vrednost pravilno napovedanih pozitivnih primerov.

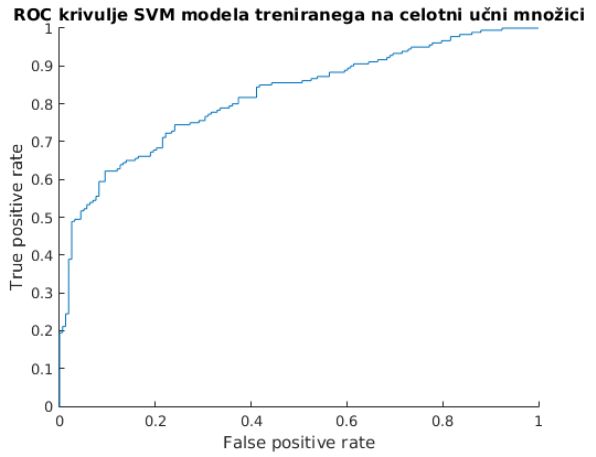
6.4 Klasifikacija z mešanim skaliranjem učne in testne množice



Slika 5: ROC krivulje SVM modelov z enako stopnjo skaliranja.



Slika 6: ROC krivulje SVM modelov z enako stopnjo skaliranja.



Slika 7: ROC krivulje realnega modela.

V zadnjem sklopu smo testirali najbolj realen primer. Pripravili smo učno množico, ki vsebuje primere iz vseh koeficientov rezanja, ki smo jih omenili v predhodnih podpoglavjih. Kot testno množico smo uporabili enako množico kot v 6.3, torej z naključnim mešanim skaliranjem.

Rezultati so predstavljeni v razpredelnici 5 in sliki 7.

Tabela 5: Realen primer.

Točnost	FNR	FPR	TPR	TNR
0.683	0.247	0.346	0.654	0.753

V približno dveh tretjinah primerov, bo naš model pravilno napovedal ali je bila slika modificirana, pri čemer je večja verjetnost, da se bo zmotil pri napovedovanju pozitivnega primera (modificirana slika) kot negativnega primera.

7. ZAKLJUČEK

V članku predstavimo več različnih pristopov, ki se uporabljajo za preverjanje integritete slikovnega gradiva. Večina pristopov je fokusirana na detekcijo modifikacij slik, ki so posledica operacije rezanja šivov, s katero lahko zmanjšamo velikost slike in hkrati ohranimo vse pomembne elemente slike. Izmed vseh pristopov je najboljše rezultate prikazoval članek, ki opisuje pristop z lokalnimi binarni vzorci, zato smo se odločili, da implementiramo njihovo idejo in delno ponovimo njihove eksperimente.

V primerjavi z rezultati iz izvornega članka, so naši rezultati za nekaj procentov slabši, vendar še vedno ekvivalentni. Sklepamo, da je do razlike v točnosti napovedi prišlo zaradi drugačnega učenja in nastavitve parametrov modela podpornih vektorjev (SVM). Pri učenju modela, so avtorji izvornega članka določene slike uporabili večkrat, medtem ko smo v naši implementaciji vsako sliko uporabili natanko enkrat. Dodali smo tudi realen primer testiranja, v katerem smo model trenirali in testirali na različnih stopnjah skaliranja, ter dosegli točnost `0.683`.

Pri analizi rezultatov smo ugotovili, da je pri nizkih stopnjah skaliranja točnost modela le za odtenek boljša od naključnega klasifikatorja, ki napove pravilni razred z verjetnostjo `50%`. Pri višjih stopnjah skaliranja, je točnost mnogo večja, vendar so ponavadi v tovrstnih primerih artefakti na sliki vidni tudi s prostim očesoma, tako da bi lahko slike klasificirali tudi brez uporabe klasifikatorja.

Poleg opisa pristopa je na voljo tudi vsa naša izvorna koda in testni primeri na javnem Github repozitoriju[1].

8. REFERENCES

[1] LIBSVM a library for support vector machines. https://www.csie.ntu.edu.tw/~cjlin/libsvm/. Accessed: 2016-04-30.

[2] Seam carving GUI program za odstranjevanje objektov s pomočjo rezanje šivov. https://code.google.com/archive/p/seam-carving-gui/. Accessed: 2016-04-30.

[3] W. L. Chang, T. K. Shih, and H. H. Hsu. Detection of seam carving in jpeg images. In *Awareness Science and Technology and Ubi-Media Computing (iCAST-UMEDIA), 2013 International Joint Conference on*, pages 632–638, Nov 2013.

[4] C. Fillion and G. Sharma. Detecting content adaptive scaling of images for forensic applications, 2010.

[5] W. Lu and M. Wu. Seam carving estimation using forensic hash. *Proceedings of the thirteenth ACM multimedia workshop on Multimedia and security - MM&Sec '11*, page 9, 2011.

[6] W. Luo, Z. Qu, J. Huang, and G. Qiu. A novel method for detecting cropped and recompressed image block. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 2, pages II.217–220, April 2007.

[7] T. Ojala, M. Pietikainen, and T. Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987, Jul 2002.

[8] T. Ojala, M. Pietikäinen, and D. Harwood. A comparative study of texture measures with classification based on featured distributions. *Pattern Recognition*, 29(1):51 – 59, 1996.

[9] A. C. Popescu and H. Farid. Exposing digital forgeries by detecting traces of resampling. *IEEE Transactions on Signal Processing*, 53(2):758–767, Feb 2005.

[10] S.-J. RYU, H.-Y. LEE, and H.-K. LEE. Detecting trace of seam carving for forensic analysis. *IEICE Transactions on Information and Systems*, E97.D(5):1304–1311, 2014.

[11] G. Schaefer and M. Stich. Ucid - an uncompressed colour image database. In *In Storage and Retrieval Methods and Applications for Multimedia 2004, volume 5307 of Proceedings of SPIE*, pages 472–480, 2004.

[12] K. Wattanachote, T. Shih, W.-L. Chang, and H.-H. Chang. Tamper Detection of JPEG Image Due to Seam Modifications. *Information Forensics and Security, IEEE Transactions on*, 10(12):2477–2491, 2015.

[13] T. Yin, G. Yang, L. Li, D. Zhang, and X. Sun. Detecting seam carving based image resizing using local binary patterns. *Computers & Security*, 55:130–141, 2015.

[1] https://goo.gl/LAHMOa

JPEG steganografija

Jon Muhovič
Fakulteta za računalništvo in informatiko
Univerza v Ljubljani

Rok Rupnik
Fakulteta za računalništvo in informatiko
Univerza v Ljubljani

POVZETEK

V seminarski nalogi orišemo področje steganografije, kratko opišemo JPEG kompresijo in naredimo pregled state-of-the-art metod, ki skušajo statistično neopazno skriti kar se da dolga sporočila v digitalne slike, stisnjene s postopkom JPEG, ena od njih je opisana v članku [8].

1. Uvod

Obravnavani članek predstavi nov pristop k vstavljanju skritih podatkov v JPEG slike. Predlagana metoda, UERD (uniform embedding revisited distortion), temelji na predhodnem delu istih avtorjev, ki so leta 2012 predlagali metodo UED (uniform embedding distortion) [7]. Oba algoritma sodita med metode, ki za izvedbo steganografije uporabljajo distorzijske funkcije. Temeljita na ideji, da z lahko z uniformno porazdelitvijo spremenjenih bitov po histogramu DCT koeficientov uspešno minimiziramo distorzijsko funkcijo in tako zakrijemo vpliv sprememb, ki jih povzroči vstavljanje informacij.

2. Pregled področja

V tem poglavju orišemo glavne elemente, s katerimi mora biti bralec članka seznanjen. Opišemo, kaj je steganografija, za kakšne namene se uporablja in zakaj je JPEG primeren format za vstavljanje skritih sporočil. Na kratko opišemo JPEG format, saj je razumevanje njegovih korakov nujno za razumevanje naprednejših steganografskih metod. Predstavimo tudi steganalizo (*steganalysis*), ki skuša v krovnem mediju (angl. cover medium) odkriti skrita sporočila.

2.1 Steganografija

Steganografija je skrivanje datotek, sporočil, slik ali drugih podatkov znotraj druge datoteke ali sporočila. Ime izhaja iz starogrških besed στεγανός (skrit, pokrit) in γράφειν (pisati). V zgodovini se je uporabljala v obliki nevidnih črnil, sporočil, skritih pod pisemskimi znamkami ipd. V digitalni obliki pa lahko sporočila prenašajo najmanj signifikantni biti slikovnih ali avdio datotek, v redundantnih delih stisnjenih datotek ali večje kriptirane datoteke, kamor neopazno skrijemo kriptirano sporočilo.

Digitalna steganografija se fokusira na statistično nezaznavnost (angl. statistical undetectability). Le-to mora uravnovesiti s količino podatkov, ki jih lahko skrijemo, torej če hočemo porabiti večji del kapacitete prenosnega medija tvegamo večjo statistično zaznavnost.

2.2 Steganaliza

Steganaliza (angl. steganalysis) je postopek, ki skuša z različnimi statističnimi metodami določiti ali neka JPEG slika vsebuje skrito sporočilo ali ne.

Steganaliza se izvaja s statistično analizo slike. Če tisti, ki je izvajal steganografijo, ni bil dovolj previden, se to lahko opazi v statističnem profilu slike. Namen steganalize ni pridobitev skritega sporočila, ampak le določitev, ali tako sporočilo obstaja. Ker se pri določitvi zaporedja spremenjenih pikslov slike uporabljajo razprševalne funkcije in ker lahko skrito sporočilo pred vstavljanjem tudi kriptiramo bi bil problem dejanskega pridobivanja skritega sporočila enostavno prezahteven. Ideja steganografije je torej, da zunanji opazovalec (v literaturi včasih imenovan *warden*) ne more razločiti med sporočili, ki vsebujejo skrito vsebino in tistimi, ki je ne vsebujejo.

Kot dataset se za testiranje steganografskih algoritmov uporablja baza slik mednarodnega izziva BOSS (break our steganographic system) [1]. Ta vsebuje 10000 nekompresiranih *bmp* slik, ki jih algoritmi lahko uporabijo kot vir *side-information*. V praksi pa avtorji člankov ponavadi sami zasnujejo svoj sistem za detekcijo spremenjenih slik. Večinoma so to klasifikatorji, ki temeljijo na poznanih metodah za strojno učenje (SVM), ki se naučijo na množici slik s sporočili in brez njih ali pa algoritmi, ki slike klasificirajo glede na njihovo ustreznost nekemu modelu. V nek delež slik skrijejo sporočila s svojim algoritmom in množico slik razdelijo na učne in testne primere. Na učnih primerih se nato klasifikator lahko nauči razlikovati med spremnjenimi in nespremenjenimi slikami. Tukaj se kaže pomanjkanje enovitega ogrodja za testiranje steganografskih algoritmov, zato je primerjanje različnih algoritmov nekoliko težavno in nezanesljivo. Izjema je recimo članek, katerega avtorji so z istim klasifikatorjem testirali več steganografskih algoritmov [6].

Steganografski algoritmi se skušajo pred staganalizo zašči-

titi na različne načine. Nekateri skušajo ohraniti statistike prvega in drugega reda (angl. statistical restoration), ohranjati števila DCT koefcientov (recimo F5 - [10]) ali minimizirati skupno spremembo DCT koeficientov z distorzijskimi funkcijami.

2.3 JPEG kompresija

Krajši opis kako deluje, saj je razumevanje potrebno za obdelavo članka in področja.

JPEG je postopek za izgubno kompresijo digitalnih slik, ki uspešno zavrže informacije v sliki, ki jih človeško oko skoraj ne zazna. Rezultat so precej manjše slike, ki (ob zmerni ravni kompresije) izgledajo skoraj enako kot originalne slike.

V postopku se slika iz barvnega prostora RGB najprej pretvori v barvni prostop YCbCr, kjer je Y intenziteta, Cb in Cr pa sta barvni vrednosti posameznega piksla (moder in rdeč kanal). Barvna kanala se lahko na tej točki skrčita (angl. downsample), saj človeško oko precej bolje zaznava razlike v intenziteti kot v barvi.

Vsak od kanalov se nato razdeli v bloke velikosti 8x8 pikslov. Če katera od dimenzij slike ni deljiva z 8, se lahko robni piksli kopirajo naprej, dokler ne dosežemo ustrezne velikosti.

Glavni korak postopka nastopi na tej točki, ko se vsak od blokov iz prejšnjega koraka z diskretno kosinusno transformacijo (DCT - discrete cosine transform) pretvori v frekvenčni prostor. DCT je nekoliko podoben Fourierjevi transformaciji, le da za opis vrednosti v frekvenčnem prostoru uporablja samo kosinuse in da so vsi koeficienti realna števila. Vrednosti v vsakem pikslu bloka se najprej centrirajo okrog 0 (torej niso več na intervalu $[0, 255]$ ampak na intervalu $[-128, 127]$ kar nekoliko poenostavi računanje DCT). Zatem se izvede dvodimenzionalen DCT na vsakem od blokov (to pomeni enodimenzionalen DCT na vsaki od vrstic in nato enodimenzionalen DCT na vsakem od stolpcev rezultata), vrednosti se tudi normalizirajo. V levem zgornjem kotu bloka dobimo opazno večjo vrednost, ki predstavlja t.i. DC koeficient (imenovan tudi konstantna komponenta), ostali elementi pa predstavljajo AC koeficiente, torej intenzitete elementov z višjimi frekvencami. Sledi kvantizacija koeficientov, kjer se vsak od elementov deli z istoležnim elementom v kvantizacijski tabeli, shranjeni na začetku JPEG datoteke in zaokroži k najbližjemu celemu številu. Kvantizacijska tabela je sestavljena tako, da daje večjo težo komponentam z nižjimi frekvencami, kar po zaokrožitvi povzroči, da veliko koeficientov za višje frekvence pade na 0.

Sledi entropijsko kodiranje vsakega izmed blokov, ki najprej komponente uredi v cik-cak vzorcu, ki gre od zgornjega levega kota v spodnji desni kot. To povzroči, da se večina elementov, katerih vrednost je enaka 0 znajde skupaj, kar izkoristi RLE (run length encoding), ki zaporedne enake elemente zapiše krajše. Na rezultatu se nato izvede Huffmanovo kodiranje, ki vrednosti še dodatno brezizgubno skrči.

3. Različni pristopi k JPEG steganografiji

Ker je JPEG najpogosteje uporabljan format za slike, je hitro pridobil pozornost steganografske skupnosti. Na tej točki je treba omeniti, kje v postopku JPEG kodiranja se sploh izvajajo steganografski postopki. Bite skritega sporočila se da skriti v kvantizirane DCT koeficiente. Različne metode se pri tem razlikujejo glede na to, katere izmed koeficientov modificirajo (nekatere uporabljajo le neničelne AC koeficiente, druge pa delujejo na vseh koeficientih, tudi tistih, ki so enaki 0 [8]) Eden prvih postopkov je bil JSteg, ki je sporočilo enostavno skril v najmanj pomemben bit (LSB, least significant bit) DCT koeficientov, različnih od 0 in 1. A taki postopki so zelo očitni že pri kratkih sporočilih, saj povzročijo artefakte v statistikah prvega reda (histogrami DCT koeficientov), ki se jih enostavno zazna. F5 [10] je algoritem, ki do problema pristopi na bolj premišljen način in absolutno vrednost koeficientov zmanjša za 1 in z nekaterimi dodatnimi triki doseže skoraj neopazne artefakte v histogramu. Poleg tega uporablja t.i. matrix embedding, ki precej poveča učinkovitost vstavljanja (t.j. da spremenimo le nujno potrebne bite, oz. da lahko zapišemo več kot en bit sporočila z eno spremembo DCT koeficienta).

V zadnjem času je precej pozornosti prejela metoda minimalne distorzije (angl. minimal distortion embedding), ki vsakemu koeficientu določi skalarno vrednost, ki določa težo (magnitudo), ki bi jo sprememba tega koeficienta vnesla v celoten postopek [7, 2, 9]. Algoritmi, ki temeljijo na tej metodi torej definirajo t.i. distorzijsko funkcijo, ki jo nato na različne načine minimizirajo in tako za vsako razmerje vstavljanja (angl. embedding rate, t.j. odstotek celotne kapacitete slike, ki ga zasede sporočilo) uporabijo teoretično najmanj zaznavne koeficiente. Koncept razmerja vstavljanja je pomemben predvsem s stališča statistične nezaznavnosti, saj je le-ta obratno sorazmerna s količino vstavljenih informacij (več informacij kot vstavimo v sliko, lažje je zaznati spremembe, ki jih s tem povzročimo).

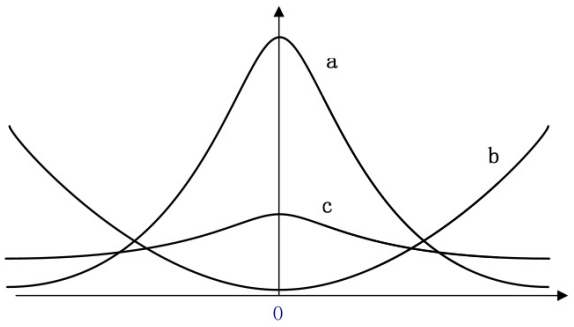
Trenutno med najboljše algoritme spadajo MOD [3], UNIWARD [9] in UERD, ki vsi uporabljajo minimizacijo distorzijske funkcije in različne načine učinkovitega kodiranja sporočila (matrix encoding, syndrome-trellis coding [4], wet paper coding).

Nekatere steganografske sheme v svojih postopkih izkoriščajo znanje o originalni sliki pred kompresijo (side-information). Originalna slika je običajno slika v visoki resoluciji, kot naprimer direktni izhod senzorja na kameri, preden je stisnjen s postopkom JPEG. V tem postopku se predpostavlja, da je originalna slika na voljo samo pošiljatelju in ne tudi zunanjemu opazovalcu (warden-u), kar daje pošiljatelju veliko prednost. V članku [5] so z uporabo *piecewise* polinomskega modela, katerega vsebina je pokvarjena z aditivnim belih šumom, dokazali, da lahko z dovolj dinamično JPEG sliko v kombinaciji z originalno sliko ustvarimo distorzijsko funkcijo, ki nam po minimizaciji vrne bolj varno skrita sporočila, kot če bi uporabili zgolj JPEG sliko. Metode z informacijo o originalni sliki dosegajo 4-krat manjšo Fisherjevo informacijsko vrednost od navadnih metod, ki spreminjajo najmanj pomembne bite DCT koeficientov v JPEG formatu.

4. Povzetek članka

Obravnavani članek gradi na prejšnjem članku istih avtorjev [7], ki predlaga metodologijo UED (uniform embedding distortion). UED skuša spremembe pri vstavljanju sporočila porazdeliti kar se da uniformno in tako zmanjšati statistično

zaznavnost sprememb, saj minimizira povprečne spremembe statistik prvega in drugega reda (to so histogrami in sopojavitvene matrike).



Slika 1: Ideja porazdelitve za uniformno vstavljanje.

V Sliki 1 krivulja a prikazuje porazdelitev DCT koeficientov (Laplaceova porazdelitev), krivulja c željeno porazdelitev spremenjenih koeficientov, krivulja b pa funkcijo, ki določa verjetnost izbire posameznega koeficienta, da doseženo kar se da uniformno porazdelitev vnešenih sprememb. V članku avtorji predstavijo *Generalized Uniform Embedding Strategy*, ki izkorišča dejstvo, da se naravnih slik ne da zanesljivo modelirati (steganografska analiza ponavadi deluje zaradi odstopanja od modela, kadar v sliko vstavimo dodatne informacije). Spremembe DCT koeficientov pa niso statistično enako zaznavne, kar pomeni, da je treba izbrati tiste, katerih sprememba bo v sliko vnesla najmanj opazno napako. Avtorji za izbiro primernih koeficientov izhajajo iz mere CV (*coefficient of variation*), ki je definirana kot: $CV(x) = \frac{\sigma(x)}{\mu(x)}$, kjer sta $\mu(x)$ povprečna vrednost DCT koeficientov, $\sigma(x)$ pa njihova standardna deviacija, kot prikazuje Slika 2. Avtorji iz tega sklenejo, da verjetnost za zaznavo spremembe pada z $|x|$ in da uniformna porazdelitev sprememb po koeficientih še ne implicira, da so spremembe v koeficientih enako zaznavne. Zato se pri vstavljanju osredotočijo na kontrolo relativne spremembe vsake od celic histograma.

V začetku so se steganografski algoritmi skušali izogniti dodajanju novih neničelnih AC koeficientov, saj to povzroči enostavno zaznavne spremembe v histogramu DCT koeficientov. Avtorji metode UERD pa ugotavljajo, da s pravilno distorzijsko funkcijo lahko informacije vstavljamo v kateri koli DCT koeficient, glede na njegovo CV utež. To pomeni, da so lahko DC koeficienti (njih se ponavadi ne uporablja za vstavljanje informacij), ki se nahajajo v regijah z visokofrekvenčnimi teksturami bolj primerni za vstavljanje kot nekateri AC koeficienti v regijah z nižjefrekvenčnimi teksturami (imajo torej večji CV).

UERD za sestavo distorzijske funkcije uporabi statistične lastnosti prvega in drugega reda za posamezen DCT koeficient. Statistika prvega reda pomeni upoštevanje frekvence posameznega AC koeficienta, torej, da se z višanjem frekvence AC koeficientov njihov CV niža (število neničelnih AC koeficientov se povečuje, verjetnost za njihovo spremembo pa zmanjšuje). Statistika drugega reda pa se osredotoča na kompleksnost teksture celotnega bloka. Le-ta se izmeri glede na količino neničelnih AC koeficientov v bloku, to pa določa težo AC koeficientov v distorzijski funkciji (koeficienti v nizkofrekvenčnih regijah imajo manjšo verjetnost za modifikacijo).

Z uporabo deljenega ključa (angl. shared key, ključ ki ga posedujeta pošiljatelj in naslovnik) se AC koeficienti najprej premešajo, nato pa se sporočilo zakodira z uporabo STC (syndrome-trellis coding), kot je opisano v [7]. Pri tem se teža posameznega koeficienta, kot jo definira distorzijska funkcija uporabi znotraj STC. Rezultat STC lahko nato apliciramo na kvantizirane DCT koeficiente, jih inverzno premešamo in nadaljujemo s postopkom JPEG. Tako dobimo končni rezultat steganografskega postopka.

5. Rezultati

Ker implementacija UERD ni dostopna, je v tem poglavju uporabljena demonstracijska implementacija algoritma UNIWARD [9].

Na Sliki 3 vidimo originalno sliko in sliko z vstavljenim sporočilom. S prostim očesom razlike seveda ne opazimo, zato pa lahko na Sliki 4 vidimo razlike koeficienotv v JPEG blokih, na Sliki 5 pa dejanske razlike v intenzitetah pikslov obeh slik. Enostavno lahko opazimo, da uporaba minimizacije distorzijske funkcije povzroči, da se sporočilo shrani v visoko teksturirane regije slike, saj so tam vrednosti pikslov veliko manj predvidljive, to pa povzroči, da so spremembe v teh regijah manj opazne.

6. Reference

[1] P. Bas, T. Filler, and T. Pevnỳ. "break our steganographic system": The ins and outs of organizing boss. In *Information Hiding*, pages 59–70. Springer, 2011.

[2] T. Filler and J. Fridrich. Design of adaptive steganographic schemes for digital images. In *IS&T/SPIE Electronic Imaging*, pages 78800F–78800F. International Society for Optics and Photonics, 2011.

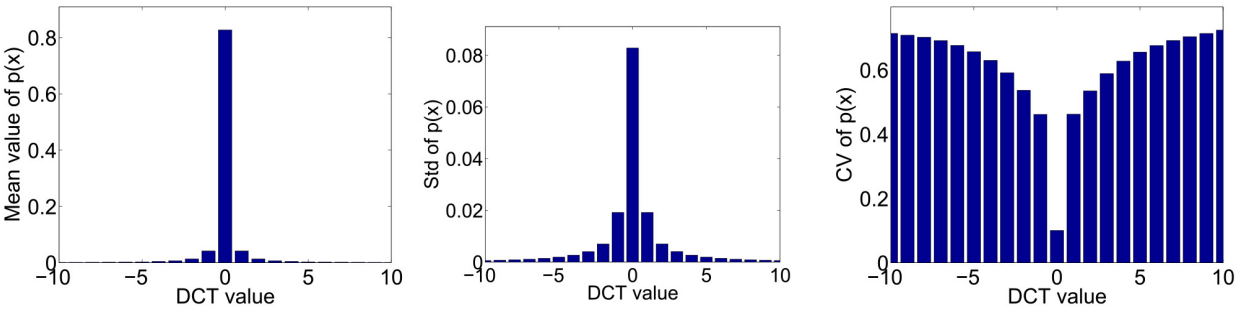
[3] T. Filler and J. Fridrich. Design of adaptive steganographic schemes for digital images. In *IS&T/SPIE Electronic Imaging*, pages 78800F–78800F. International Society for Optics and Photonics, 2011.

[4] T. Filler, J. Judas, and J. Fridrich. Minimizing additive distortion in steganography using syndrome-trellis codes. *Trans. Info. For. Sec.*, 6(3):920–935, Sept. 2011.

[5] J. Fridrich. On the role of side information in steganography in empirical covers. In *IS&T/SPIE Electronic Imaging*, pages 86650I–86650I. International Society for Optics and Photonics, 2013.

[6] J. Fridrich, T. Pevný, and J. Kodovský. Statistically undetectable jpeg steganography: Dead ends challenges, and opportunities. In *Proceedings of the 9th Workshop on Multimedia & Security*, MM&Sec '07, pages 3–14, New York, NY, USA, 2007. ACM.

[7] L. Guo, J. Ni, and Y. Q. Shi. Uniform embedding for efficient jpeg steganography. *IEEE Transactions on Information Forensics and Security*, 9(5):814–825, May 2014.



Slika 2: Histogrami DCT koeficientov.

(a) Originalna slika

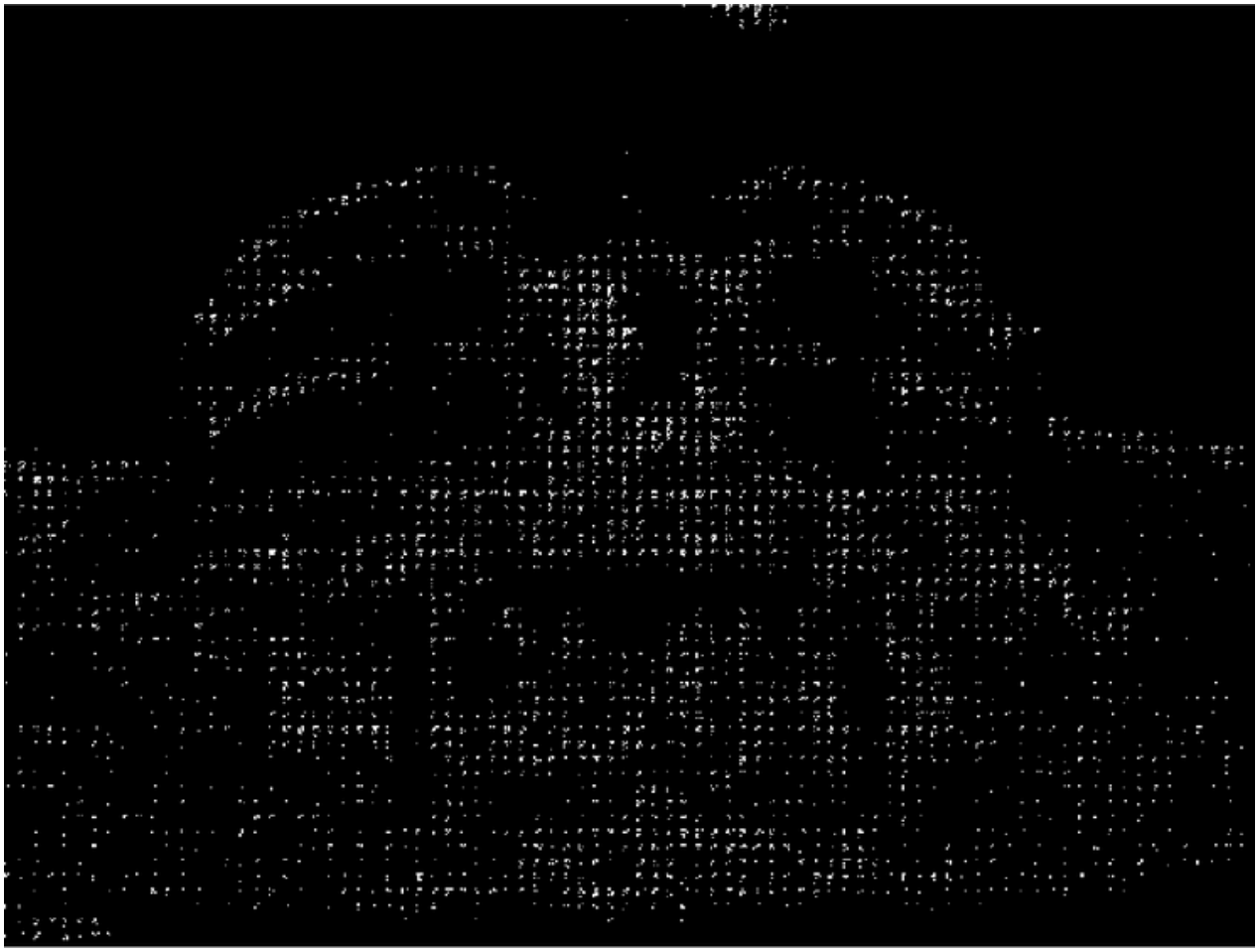
(b) Slika s skritim sporočilom.

Slika 3: Slika s skritim sporočilom.

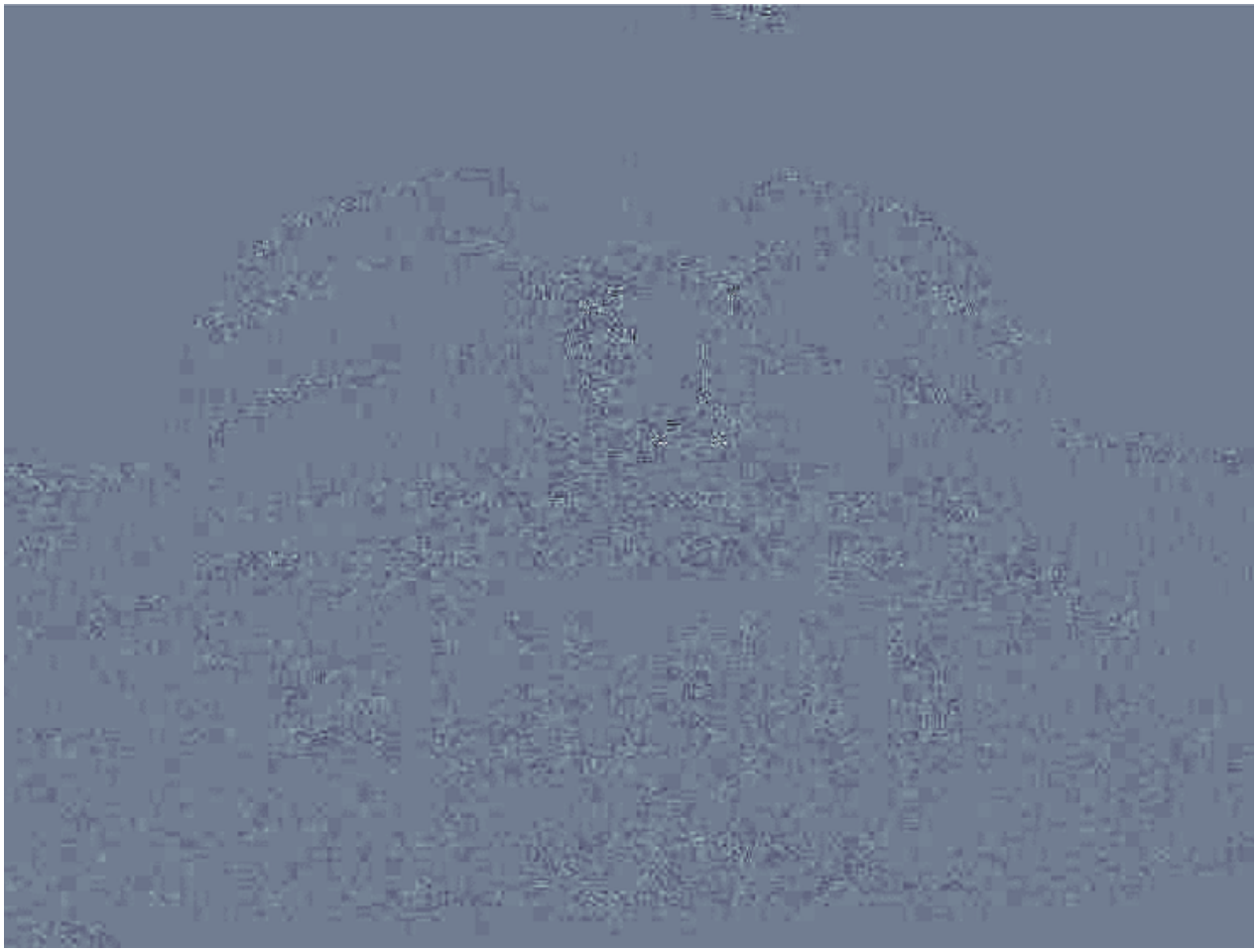
[8] L. Guo, J. Ni, W. Su, C. Tang, and Y. Q. Shi. Using statistical image model for jpeg steganography: Uniform embedding revisited. *IEEE Transactions on Information Forensics and Security*, 10(12):2669–2680, Dec 2015.

[9] V. Holub, J. Fridrich, and T. Denemark. Universal distortion function for steganography in an arbitrary domain. *EURASIP Journal on Information Security*, 2014(1):1–13, 2014.

[10] H. C. D. B. Steganalysis and A. Westfeld. F5—a steganographic algorithm. In *Information Hiding: 4th International Workshop, IH 2001, Pittsburgh, PA, USA, April 25-27, 2001. Proceedings*, volume 2137, page 289. Springer Science & Business Media, 2001.



Slika 4: Razlika v DCT prostoru.



Slika 5: Razlika v intenzitetah pisklov.

Del IV

Forenzika podatkovij in triaža

Overview of the Tiered forensic methodology model for Digital Field Triage by non-digital evidence specialists

Jure Kolenko
FRI UL
jk6789@student.fri.uni-lj.si

Matej Danicek
FRI UL
danicma1@uhk.cz

ABSTRACT

This paper provides an overview of the "Tiered forensic methodology model for Digital Field Triage by non-digital evidence specialists"[1] paper by Ben Hitchcock, Nhien-An Le-Khac, and Mark Scanlon presented on the DFRWS EU conference on 31.03.2016 [1], as well as a general overview of the digital forensic triage and investigation. The paper deals with new ways (model) to conduct the forensics investigations with focus on budgetary and time (laboratory backlog) constraints. The main proposed change consists of elementary training of front-line personnel in the field of digital triage conducted on-scene or shortly after the evidence collection phase. The paper also presents an overview of implementation of this model in real-world environment (namely Canadian police forces).

Keywords

digital forensics, triage, digital investigation

1. INTRODUCTION

As the society uses more and more digital devices (including computers, tablets, smart-phones, wearable technology, etc.) this is reflected in their part in criminal activities. This results in increasing demand in the field of digital forensic examinations and in turn in prolongation of backlogs and delays in Technological Crime Units (TCUs) investigations. This leads to investigators not having potentially actionable leads and information at the appropriate times. According to James and Gladyshev, in 2014 the backlogs in the USA range from 12 to 18 months in some cases.[2]

The issue with forensic personnel attending the search scene and conducting an on-site analysis lies in backgrounding their regular file examination duties in the forensic laboratory, since in-field analysis has now the highest priority no matter what type, severity or time sensitivity of the related crime. Since fully training a forensic analyst is quite time and money consuming, the paper proposes a process model with personnel trained in the basics of forensic analysis to be included during the on-site triage and initial examination phases of the digital investigation process. The aim of the paper was to provide a framework and guidelines for specifically trained personnel in the field of digital field triage and in turn reduce the load on TCUs and consequently their backlog.

[1]The paper is available from the sciencedirect.com server (http://www.sciencedirect.com) and was published by Elsevier Ltd on behalf of DFRWS under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

2. FORENSIC TRIAGE

The need for forensic triage has emerged with more and more technology being used by the society in general and subsequently the criminals. Providing actionable evidence or leads in a timely manner became one of the most important outcomes of the forensic examination.

2.1 Background

As Rogers et al. [3] mentions, there have been number of investigative models proposed to be applied in the forensic examination field. However, all these models assumed that all digital media (or their images) would be transported to a dedicated forensic examination laboratory and a whole and thorough analysis would be conducted there. Using this approach might be extremely time-consuming and thus not being efficiently applicable in a time-critical situation where not obtaining a piece of digital evidence or a lead in a timely manner might have grave consequences (for example child abductions, death threats, person of interest localization etc.). In these cases the need for quickly obtaining potentially useful information precedes the need for full in-depth analysis.

The Computer Forensics Field Triage Process Model has been developed as a formalization of new investigative methods used in real investigations by agents co-operating with Assistant U.S. Attorney Steve Debrota's office. In several cases, the office has been involved in investigations where the results of a quick and efficient examination (conducted on-site) have played a major part in securing the offender's conviction and protecting possible future victims. The formalization of the process took place after the attorney's office has reached out to the Computer and Information Technology Department at Purdue University and the National White Collar Crime Center in order to articulate and struc-

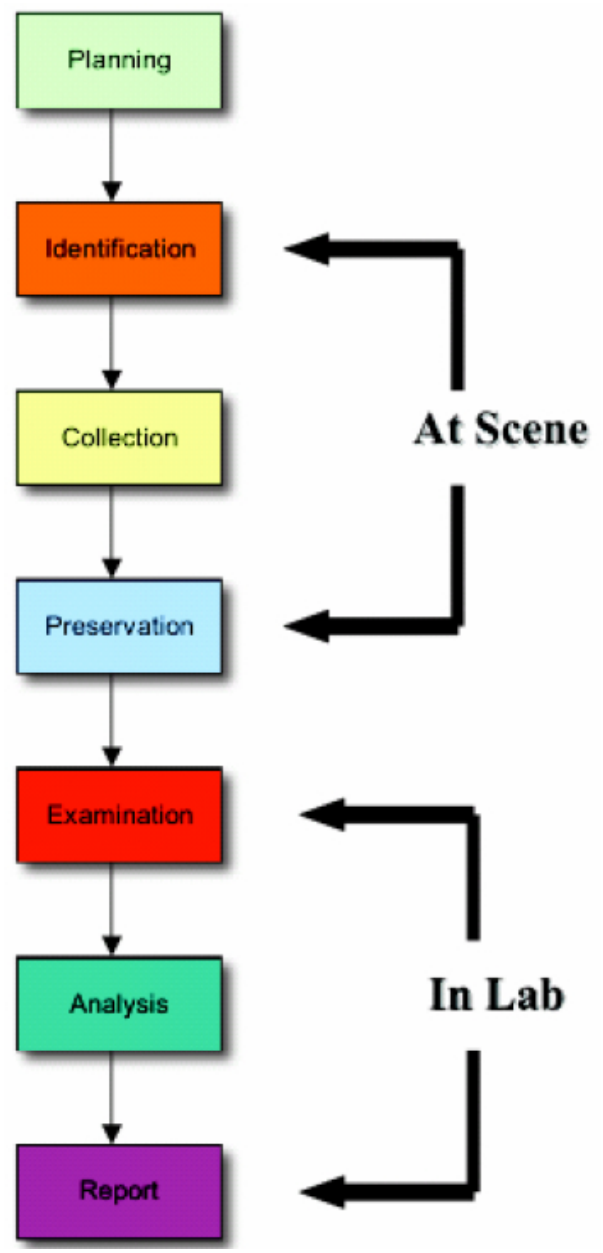


Figure 1: Traditional digital forensic examination process models. [3]

ture a formal model that could be replicated in other law enforcement agencies and related institutions. The model has been evaluated with positive feedback by 20 State and Local Law Enforcement Officers during a seminary that took place at Purdue University during the fall of 2005.

The model focuses mainly on:

1. Finding usable evidence immediately;
2. Identifying victims at acute risk;
3. Guiding the ongoing investigation;
4. Identifying potential charges; and
5. Accurately assessing the offender's danger to society.

Additionally the speed with which relevant information can be obtained directly provides a psychological advantage during suspect interrogation by the investigators, where the suspect is more likely to be cooperative when immediately faced with the obtained evidence as well as possible "triggers" which the interrogator can use to steer the suspect into cooperating or revealing addition information about his/her involvement.

2.2 Phases of the CFFTPM

The phases of the CFFTPM are expanding the ones from Carrier's and Spafford's Integrated Digital Investigation Process model (IDIP) and the Digital Crime Scene Analysis (DCSA) model by Rogers. These phases (described in Figure 2) include: planning, triage, usage/user profiles, chronology/timeline, internet activity, and case specific evidence.

2.2.1 Planning

First is the planning phase during which the types of actions to be taken, circumstances of the investigation, number and types of devices to examine and other aspects of the examination are considered and planned for. The situation can be characterized by the acronym SALUTE:

1. Strength - how many persons of interest are involved and possibly what is the level of their technological knowledge
2. Activity - potential actions, related to the investigated crime, the suspect(s) could have taken
3. Location - both physical and cyberspace environment which is the focus of the investigation
4. Uniform - clear markings, symbols, or corporate or agency identifiers; including email addresses, usernames, network domains, etc.
5. Time - chronological plan of the investigation based on previously gathered case details
6. Equipment - what types of devices and software can be expected

With the information contained in the SALUTE the identification of possible approaches, number of needed personnel, types of hardware and software needed and other key properties can be assessed and appropriate decisions can be made.

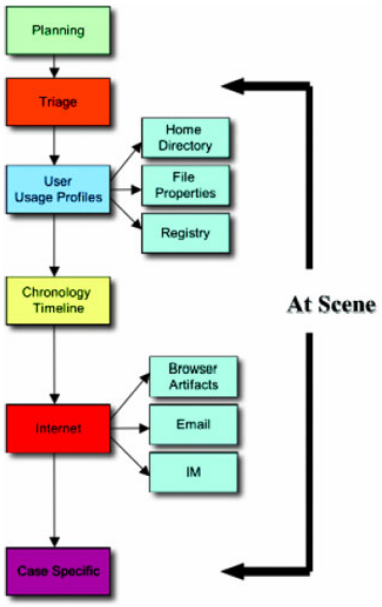


Figure 2: Phases of the CFFTPM.[3]

2.2.2 *Triage*

Once the scene has been properly secured and is controlled, the triage phase can begin. Similarly to medical triage on disaster sites, the main goal is to prioritize and make use of the limited resources (time, personnel, etc.) with maximally beneficial results.

The volatile media or evidence containers (such as RAM, routing tables, DHCP lease tables, etc.) have priority over the relatively stable ones. Other prioritization criteria would be the relevance to the case and time it would take to obtain relevant evidence from the media.

At this point, the investigators and interviewers dealing directly with the suspect provide information to the examiners in order to steer the triage process in the most efficient direction.

2.2.3 *Usage/User Profiles*

After the prioritization during the triage phase, the examination itself begins. A clear link between a piece of evidence and one particular and identifiable suspect must be made. In some cases (for instance when one user account is being shared by multiple users or is publicly accessible) providing this link may be difficult and thus the effort conducting a deep analysis of the particular user profile may not yield expected results.

Notable areas that can provide usable evidence may include the user's home directory (and its notable subfolders such as Desktop, Pictures, Documents, etc.), file properties (permissions can shed light onto which user account is associated with the given file and to some extent provide the link with the particular suspect) and the registry in case of Windows systems.

2.2.4 *Chronology/Timeline*

When composing the timeline of supsect's actions numerous sources can provide relevant data. The MAC times (modified/accessed/created timestamps) can be used (alas some possible inconsistencies and shortcomings must be considered), as well as system and application logs. It might also be useful to look for irregular time periods - those with properties deviating from the usual or expected ones. For instance unexpectedly missing log entries indicating possible tampering.

2.2.5 *Internet*

In this phase the user's activities on the internet are the focus of the examination. Almost all cases require at least some level of examination of the internet activities artifacts although their importance and acquirement efficiency (considered with regards to the time cost) will vary significantly between cases.

The usual areas providing these artifacts include:

1. Browser artifacts - including "cookies", web pages history, and cached files
2. Emails - email artifacts can provide many useful pieces of evidence, however, their acquirement and filtering may be time consuming
3. Instant Messaging - IM clients may store conversations and contact information locally and therefore can be subjected to forensic examination, however, their examination can require going through large amounts of data and thus be very time consuming unless the examiner is looking for something specific (in which case using string search tools is recommended)

2.2.6 *Case Specific Evidence*

During the examination, its focus can change based on the case specifics. The examiner needs to evaluate resources, make use of known specifics and prioritize as well as customize search goals. One consideration when prioritizing examination goals and directions is whether the evidence collection is time bounded (after some point the collection is to be stopped) or unbounded. As time is of the essence in both of these situations, in case of the unbounded investigation taking some more time-consuming directions may be justifiable.

3. DIGITAL FIELD TRIAGE MODEL (DFT)

The Digital Field Triage Model proposed by Hitchcock et al.[1] is loosely based on and expands the CFFTPM with the main difference of considering a DFT (with limited qualifications) member for conducting the on-scene analysis instead of a fully qualified forensic analyst.

3.1 Planning

In the planning stage of the investigation the DFT member provides assistance to the case investigators and, similarly to the SALUTE concept, conducts a risk assessment including answering questions like:

- Is the device mission-critical and cannot have any downtime?

Figure 3: Phases of the DFT model.[1]

- Skill-set of the DFT member - does he/she posses the knowledge to efficiently conduct the analysis?
- What are the suspect's abilities?
- What type of crime is being investigated?

Since the DFT member is a part of the investigation from the beginning, he/she should have all the necessary information to answer all these questions and act in the most appropriate way.

3.2 Assessment

During the assessment phase the DFT member identifies the digital evidence and conducts an on-scene triage (selection and prioritization) after which he can use the TCU approved tool and methodology to assess and examine the digital media.

3.3 Reporting

After completing the assessment, in order to bring the investigator's attention to the important artifacts, the DFT member creates an Observation Report containing a list of case-relevant and notable artifacts that have been obtained. In a case of identity theft this report might contain, for example, passport-style photographs and digital ID document templates.

3.4 Threshold

After the reporting phase the DFT member and the investigator need to determine which of the artifacts extracted meet the threshold for further in-depth analysis by the TCU. The correct distinction relies on the TCU approved tool and methodology as well as the DFT member's training and experience.

4. IMPLEMENTATION

The first model was implemented in 2009. It consisted of a parent TCU with 25 members and 8500 employees from Federal, Provincial and Municipal police forces. The DFT model was divided into digital computer (DCFT) and digital mobile field triage (DMFT) lines.

4.1 TCU approved tool

The policy required DFT members in the DCFT line to interrupt the boot process in order to conduct the investigation. Commercial tools were considered, but were deemed not viable because of the cost involved. Many open-source and commercial tools were also not viable because they were geared towards TCU members and so too complicated. The decision was made to build a custom Ubuntu disk, maintained by the parent TCU.

For the DMFT line, the primary concern was compatibility with different mobile devices, so a commercial tool was chosen.

There is a need for periodic surveys of the tools chosen, to ensure the best tool for the job is being used.

4.2 Training

The candidates were selected based on a questionnaire, which covered investigative and computer experience, and on needs of the parent TCU, such as location and availability of the candidates.

The Digital computer field triage course is five days long and takes a hands on approach. The first day covers the expectations of their participation, how to handle digital evidence, legal considerations and investigational techniques. The second day provides training on how to interrupt the boot process and the capabilities of the TCU approved tool. This was found to be one of the hardest parts of the course. The candidates then practice searching for evidence, and the final topic is an explanation of the observation report. On day three and four, the candidates work at their own pace. They are provided with several scenarios, similar to those they will be asked to assist on, on which they must perform investigations and finally produce an observation report. The reports are reviewed and the candidates appropriate receive feedback. The last day is testing day. The candidates receive a scenario, which they must investigate on their own and then produce an observation report. After they complete the report, they are also given a test covering the materials of the course. The candidates that pass are then certified as DCFT members.

The Digital mobile field triage course is four days long and, as before, takes a hands on approach. Day one is lecture based, but the candidates are still familiarized with use of the approved tool. Days two and three are practical and consist of exercises for different mobile operating systems, with assistance given from instructors. On the last day, the candidates must investigate scenarios on their own and answer a series of questions. The candidates deemed to pass are certifiefd as DMFT members.

4.3 Continuing education

Techniques and technology are constantly changing, so there needs to be a way of passing new information to DFT members. A forum has been set up so DFT members can access new materials and instructional videos.

Skills learned also deteriorate from lack of use over time, so each DFT member has to complete a minimal amount of assessments per year to maintain their qualification. If there are no cases in the area a member is in, they can request scenarios from their parent TCU to meet the requirements.

4.4 Management

The management of the program is integral to the success of the DFT model. Initially two senior analysts were assigned as a side project, but it turned out a dedicated forensic analyst was needed. The coordinator is responsible for ensuring the DFT members' competency meets appropriate standards. He reviews all observation reports for accuracy and quality, identifies training issues, and ensures only the approved tools and methods are used.

The role of the coordinator is important, because the DFT members should not work in isolation. The DFT model is not a replacement for full analysis, so the communication between DFT personnel and their TCU parent units is imperative. Even in small units, where there is no dedicated TCU, a TCU from another area must take the role of a parent.

The same server used for continuous training of DFT members is used by coordinators to track the members' assignments. The server provides case numbers for each investigation for DFT members. When a case number is assigned, the name of the person who requested it is also recorded, which provides metrics for the coordinator.

5. RESULTS

The primary objectives were to increase the efficiency of an investigation and to decrease the backlog of files for analysis.

Since the start of the program, there has been an increase in the number of guilty pleas during the investigation, because the observation reports provide evidence to prosecutors and defence counsels quickly, which makes the choice easier.

The backlog was reduced greatly. The amount of evidence files forwarded to parent TCUs decreases by 75%, which means the parent TCUs have a lot less work to do. In year 2009, when the program started, there were 522 files in the backlog, but by year 2015 there were only 137. This, however, cannot be solely attributed to the DFT program, as a more selective process of files being accepted by the TCU was put into place.

5.1 Review

The DFT members have expanded the resources available to parent TCUs which is improving the efficiency of the entire program. There has been a shift in location of assessment from the field to the office as the DFT members only assist in collecting the evidence, but don't process it on the scene. The drawback of this is that the DFT members cannot always identify additional storage, such as connected USB devices.

To ensure the integrity of the program a continuous assessment needs to be conducted on a regular basis. The DFT coordinator could do a full analysis on a randomly chosen investigation, thus providing quality assurance. The training course also needs to be evaluated. A change which was already made was to make some of the instructors experienced DFT members, who provide better insight into the tasks being conducted. Another area which could be improved is in candidate selection and better metrics for judging their abilities.

The TCU approved tool works fine, but it is custom built, which is a problem. There is limited support because of that. Academia needs to be involved to avoid the costs of commercial tools and to increase their effectiveness, trust and credibility.

The final evaluation step was determining whether the program is being used and actually wanted by the staff. There were constant requests for more courses from supervisors, which means the program was a success.

5.2 Model evaluation

The DFT model was created following the principles of Digital science research process, so should be evaluated on the following conditions:

- Is it consistent with the models in the field of digital forensics?
- Is the model usable?
- Does the model guide the handling of digital evidence?

The basis of the model were the concepts from the CFFTPM. The model is not meant to replace laboratory procedures, but aid in collecting useful evidence in a manner such that the integrity of the evidence is maintained, which makes it consistent with other models in the field.

A model is considered usable if it can be put into practice in real life and the desired outcome is achieved. Over 1800 files were assessed and while they are not perfect they do provide non-digital evidence specialists with a roadmap for dealing with evidence. It also provides extra information on what courses of action to follow, and where there are risks.

6. CONCLUSION AND FUTURE WORK

The focus was to offload some tasks from forensic analysts to non-digital evidence specialists. The digital evidence is seized, then a DFT member determines which evidence meets the threshold for further TCU examination. There is a need for DFT members to acquire the contents of Random Access Memory (RAM) from a running machine, but it requires greater training for DFT members.

The initial problem was to create a DFT model to assist in investigations and decrease the backlog of digital evidence. By review it was found that the new model met the requirements, with increase in investigational efficiency being the most beneficial.

Investigators are receiving actionable information in a timely matter, as opposed to reacting to evidence obtained months after the crime has happened. This shows the DFT program is an integral part of investigations involving digital evidence. The DFT program relies on TCU to provide oversight and training and the TCU relies on the DFT program

to provide actionable information. The long term goal is every piece of evidence being examined before it is sent to the TCU for further examination, which means less items of work for the TCU, so they are able to spend more time on each item.

While the DFT model described is useful, there are still a number of possible improvements:

- Advanced training for experienced DFT members including RAM capture and encrypted volume detection.
- Use of experienced members as mentors and trainers.
- More accurate metrics to determine efficiency and effectiveness of the model.

There are also plans to build a virtual platform for the purpose of training non-digital forensic investigators. It will integrate digital forensic scenarios such as observing and recording criminal activity, acquiring digital evidence, live forensic tasks and others.

7. REFERENCES

[1] B. Hitchcock, N.-A. Le-Khac, and M. Scanlon. Tiered forensic methodology model for digital field triage by non-digital evidence specialists. In *DFRWS 2016 Europe - Proceedings of the Third Annual DFRWS Europe*. DFRWS, April 2016.

[2] J. James and P. Gladyshev. Automated inference of past action instances in digital investigations. *International Journal of Information Security*, 14(3), June 2015.

[3] M. Rogers, J. Goldman, R. Mislan, T. Wedge, and S. Debrota. Computer forensics field triage process model. *Journal of Digital Forensics, Security and Law*, 1(2):19–38, 2006.

Hitro forenzično slikanje velikih diskov s presejanjem

Sven Cerk
Fakulteta za računalništvo in informatiko
Univerza v Ljubljani
Ljubljana, Slovenija

Svit Timej Zebec
Fakulteta za računalništvo in informatiko
Univerza v Ljubljani
Ljubljana, Slovenija

Miha Eleršič
Fakulteta za računalništvo in informatiko
Univerza v Ljubljani
Ljubljana, Slovenija

POVZETEK

V tem članku je predstavljen postopek hitrega kloniranja diskov avtorjev Jonathana Griera in Goldena G. Richarda III. V primerjavi z obstoječimi pristopi k reševanju problemov, ki jih predstavljajo vedno večje kapacitete diskov, ta postopek zagotavlja veliko večjo zanseljivost dokaznega gradiva in ponovljivost preiskave. Rezultat slikanja diska s presejanjem je natančna nizko nivojska kopija izbranih sektorjev, shranjena v formatu *AFF v3*. Na ta način je odpravljena težava forenzike v živo, kjer izgubimo podatke z diska in shranimo le rezultate analize. Zelo pomemben je izbor sektorjev, ki je skoraj popolnoma avtomatiziran in od preiskovalca zahteva le definicijo profila preiskave, ki s preprostimi pravili usmerja iskanje pomembnih sektorjev pri zajemu. Na testnih primerih je opisani postopek dosegel več kot 3-kratne pohitritve z manj kot 5% izgube dokaznega gradiva.

Ključne besede

računalniška forenizika,triaža, delni zajem podatkov, presejanje

1. UVOD

Nenehno povevečevanje diskov postaja velik izziv računalniškim forenzikom. Problem je znan kot "izziv velikosti" in predstavlja eno izmed večjih groženj digitalni forenziki. Zajemanje podatkov z velikih diskov je počasno; slikanje (angl. imaging) celotnega diska lahko traja več kot deset ur [11, 12, 6, 10].

Obstoječ pristop za reševanje tega problema je združitev procesa zajema in analize s pomočjo forenzike v živo ali triaže. Kljub temu, da reši problem velikosti podatkov, uvede druge probleme. Prvi je, da shranimo rezultate analize in ne izvirnih podatkov, drugi pa, da onemogoči dodatno analizo po zaključeni triaži ali prvotni analizi. Zaradi teh problemov je slikanje celotnega diska še vedno najbolj uveljavljen način zajema podatkov med digitalno forenzično preiskavo.

Želeli bi zmanjšati količino podatkov za zajem in hkrati ohraniti dobre lastnosti slikanja. Metoda, ki zadosti tem pogojem, imenovana presejanje (angl. sifting collectors), je opisana v članku [7]. Osnovni princip te metode je selektivno slikanje forenzično pomembnih regij na disku.

2. ZAJEM DIGITALNIH PODATKOV

Trenutno najbolj uveljavljena metoda zajema podatkov med digitalno forenzično preiskavo je kloniranje celotnega diska. Zagotavlja nam preverljivost in ponovljivost ugotovitev, do katerih pridemo med analizo. Prav tako omogoča kasnejšo dodatno analizo novih predpostavk, saj vsebuje vse podatke izvornega diska. Težava te metode je predvsem počasen zajem. Problematično je tudi shranjevanje velikih slik diskov.

2.1 Delni zajem podatkov

Triaža in forenzika v živo zabrišeta meje med zajemom in analizo forenzničih dokazov. Računalniške naprave analiziramo sproti, shranjujemo pa le informacije relevantne za primer. To so lahko datoteke, zabeležke, zgodovina brskalnika in podobno [13]. Takšen način analize je hitrejši, vendar tako pridobljeni podatki niso preverljivi, saj so shranjeni podatki že interpretirani. V primeru datotek surove podatke z diska interpretira gonilnik datotečnega sistema, v primeru zgodovine brskalnika je v verigo dodano še orodje za branje podatkovnih baz.

Članek [7] opisuje način shranjevanja forenzičnega gradiva, ki ne shrani celotne slike diska, hkrati pa ne vpliva na ponovljivost analize.

2.2 Shranjevanje delnih slik

Namesto shranjevanja končne oblike dokaznega gradiva lahko pri analizi v živo med postokom analize shranjujemo vse dostopane sektorje v delno sliko diska. Na ta način še vedno zmanjšamo količino zajetih podatkov in s tem tudi čas zajema, vendar ne izgubimo preverljivosti. Iz delne slike diska je možno ponoviti analizo, saj smo shranili surove in neinterpretirane podatke z diska.

Posebna prednost tega pristopa je kompatibilnost z obstoječimi orodji. Sliko lahko shranimo v formatu *Advanced Forensics Format – AFF v3*, ki je pogosto uporabljen v forenzičnih preizkavah in ga podpira večina orodij [6]. S tem se izognemo dolgotranjenemu postopku uveljavitve novega formata.

Format sicer ni namenjen shranjevanju delnih slik, vendar

je zaradi njegove strukture to vseeno mogoče. Sestavljen je namreč iz več delov imenovanih segmenti. Do segmentov dostopamo preko glavnega slovarja, ki se nahaja na začetku datoteke, posamezni segmenti pa so lahko shranjeni v poljuben vrstnem redu. Vsak segment hrani majhen del diska. Če segmente, do katerih med analizo nikoli nismo dostopali, izpustimo, dobimo manjšo sliko diska. Dobljena slika vsebuje vse dostopane podatke.

Tako zgrajena datoteka ne ustreza popolnoma standardu *AFF v3*. Nekatera orodja, npr. *AccessData FTK* delujejo brez sprememb. Nekatera druga orodja pa potrebujejo manjše popravke [7]. Potrebno je definirati, kako ravnati v primeru dostopa do manjkajočih sektojev. Vračamo lahko same ničle, lahko jih označimo kot poškodovane ali pa uvedemo posebno napako, ki pove, da podatki niso bili zajeti.

Če ne želimo uporabiti formata *AFF v3*, lahko uprabimo tudi surove diskovne slike, kjer za nezajete dele uprabimo funkcije datotečnega sistema za shranjevanje delnih datotek (sparse files). Ta možnost je bolj podprta, a ima težavo, da se datoteka med kopiranjem lahko "napihne" na originalno velikost diska. Prav tako nimamo nadzora nad branjem nezajetih sektorejev, saj je to odvisno od datotečnega sistema, kjer je delna slika shranjena – običajno to pomeni ničle.

3. AVTOMATIZACIJA ZAJEMA

Pri forenziki v živo dodatno omejitev predstavlja dejstvo, da mora forenzik sproti analizirati najdeno dokazno gradivo in oceniti njegovo pomembnost, kar lahko zahteva veliko časa.

V tem razdelku je predstavljen avtomatski postopek za delni zajem podatkov. Postopek temelji na avtomatski oceni pomembnosti določenih podatkov.

3.1 Določanje forenzične pomembnosti regij

Disk, ki ga želimo zajeti, razdelimo na regije, katerim določimo pomembnost in oslikamo le regije z visoko pomembnostjo.

Preden določimo pomembnost regije, jo moramo natančno definirati.

Definicija 1. Regija je *forenzično pomembna*, če, in samo če, se rezultat povezane preiskave v primeru, da vsebino regije nadomestimo z naključnimi podatki, močno spremeni.

Kot vidimo, pomembnost ni le lastnost regije, temveč je močno vezana na kontekst forenzične preiskave. V različnih preiskavah se lahko namreč osredotočamo na povsem drugačne vrste podatkov. Pri preiskovanju gospodarskega kriminala se lahko osredotočamo na elektronsko komunikacijo, medtem ko pri preiskovanju razpečevanja zlonamerne programske kode pregledujemo datoteke z izvorno kodo.

Takšna definicija forenzične pomembnosti ni uporabna, saj moramo za njeno določitev opraviti celotno preiskavo in jo ponoviti z naključnimi podatki na mestu izbrane regije.

Malo lažje je določiti nepomembne regije, kar je zajeto v naslednji definiciji.

Definicija 2. Regija, ki tekom preiskave ni nikoli prebrana, je *očitno nepomembna*.

Obe definiciji zgoraj sta vezani na podatke, ki jih dobimo šele po opravljeni preiskavi. Mi želimo pomembnost regij določiti vnaprej in že pri zajemu izpustiti nepomembne regije ter ga tako pohitriti.

Prave pomembnosti pred zaključkom preiskave seveda ne moremo poznati. Lahko pa ocenimo verjetnost, da bo regija pomembna.

Definicija 3. Pričakovana pomembnost je verjetnost, da bo regija za dano (še neopravljeno) preiskavo forenzično pomembna.

Pričakovano pomembnost regije lahko določimo na podlagi podatkov iz preteklih preiskav. Avtorji članka [7] sklepajo, da v splošnem veljajo sledeče trditve:

Trditev 1. *Regije, ki niso bile nikoli alocirane, imajo nizko pričakovano pomembnost.*

Trditev 2. *Regije, ki vsebujejo metapodatke o diskih in datotečnih sistemih, imajo visoko pričakovano pomembnost.*

Trditev 3. *Regije, za katere vemo, da vsebujejo zanimive podatke (npr. predpomnilnik spletnega brskalnika) ali podatke, ki pripadajo znanim zanimivim lokacijam (npr. imenik beležk), imajo visoko pričakovano pomembnost.*

Trditev 4. *V splošnem je pričakovana pomembnost močno korelirana z lastnostmi, ki jih lahko razberemo iz datotečnega sistema, kot so imena datotek, tipi datotek, čas zadnjega dostopa, spremembe . . .*

V nadaljevanju opisani pristop temelji na veljavnosti teh trditev, saj pri njem slikamo le regije diska, ki imajo visoko pričakovano pomembnost.

3.2 Hitro odkrivanje pomembnih regij

Kot je opisano v razdelku 2.2 naš pristop zajema hitro preiskovanje diska in slikanje vseh obiskanih sektorjev. Preiskovanje izvedemo tako, da obiščemo le pomembne regije, v katerih se nahajajo dokazi povezani s preiskavo. Ostale regije izpustimo.

Pri ročni analizi je začetni postopek v večini primerov skoraj enak. Najprej preiščemo začetne sektorje naprave, kjer preberemo tabelo razdelkov. Glede na informacije iz tabele razdelkov preberemo začetne sektorje vsakega razdelka. Tako dobimo informacije o datotečnih sistemih. Ko poznamo datotečni sistem, se lahko po disku sprehajamo po imenikih od korenskega naprej in iščemo pomembne datoteke.

Nadaljevanje analize (izbira zanimivih imenikov, datotek . . .) pa je vedno bolj odvisno od lastnosti konkretnega primera.

```
/Outlook/
/Thunderbird/Profile/
/App.*/Windows Live Mail/
/AIMLogger/
/My Chat Logs/
/My Received Files/
```

```
\.(pst | ost | pab | mab | emx | nsf | edb)$
\.(mbox | eml | msg | xls | xlsx)$
^SAM$
^SECUR1TY$
^SOFTWARE$
^NITUSER\.DAT$
^SYSTEM$
```

Slika 1: Profil *NPSJean*.

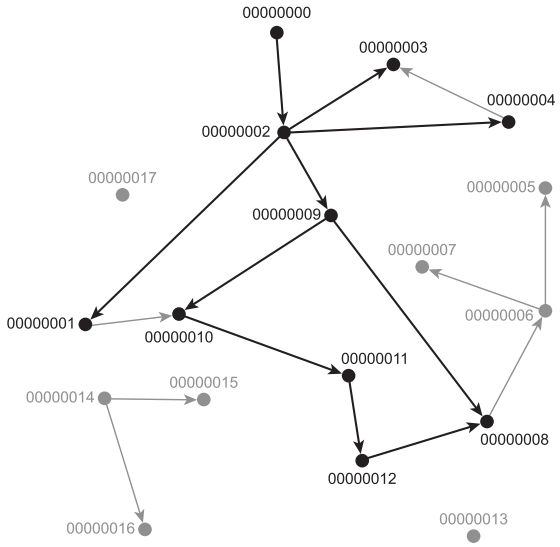
Za avtomatizacijo tega dela analize moramo opisati postopek, ki bo zmožen izbiranja med pomembnimi in nepomembnimi podatki na disku. Ta postopek temelji na trditvah opisanih v razdelku 3.1. Trditve nam povedo, da lahko pričakovano pomembnost regij ocenimo s pomočjo podatkov iz starih preiskav s podobnimi lastnostmi. V ta namen avtorji članka [7] predlagajo definiranje različnih profilov za različne primere. Preprosti profili so lahko na primer *Registry*, *IE-History* ali *Email*, ki opisujejo regije povezane z registrom, zgodovino brskalnika Internet Explorer ali elektronsko pošto.

Avtorji v članku [7] predstavijo zelo preprost jezik za opisovanje profilov, ki temelji na regularnih izrazih. Profil je zbirka regularnih izrazov, ki opisujejo lokacije na disku, za katere določimo ali jih v preiskavi želimo obravnavati ali ne. Na sliki 1 je prikazan primer profila, ki je bil uporabljen za analizo dokazov v hipotetičnem primeru *M57-Jean* [1]. Na sliki vidimo, da so v preiskovanje vključeni imeniki `/Outlook/`, `/Thunderbird/Profile/`, `/My Chat Logs/`, `/My Received Files/` in vsi imenik z imeni, ki se začnejo z nizom `/App` in končajo z `/Windows Live Mail/`. V preiskavo so vključene tudi razne datoteke s končnicami `pst`, `ost` ... Izključene pa so poti, ki vsebujejo `SAM`, `SECURITY` ...

Z izbranim profilom nato pričnemo s slikanjem diska, pri čemer se po disku premikamo po referencah med podatki in obiskujemo le regije, ki jih določa izbrani profil.

Kot je opisano v razdelku 2.2, se zaradi zagotavljanja ponovljivosti preiskave v končno sliko diska klonirajo vsi obiskani sektorji. Če želimo postopek pohitriti, moramo torej poskrbeti, da ne obiščemo vseh sektorjev naprave. To zagotovi predpostavka, da so vse pričakovano pomembne regije, ki jih določa profil, dostopne preko referenc, ki izhajajo iz korenskega imenika. S to predpostavko zahtevne operacije, kot je iskanje ključnih besed po celotnem disku, niso potrebne, saj pričakovano pomembne regije vedno najdemo na neki poti iz korenskega imenika. Takšnemu postopku — kjer selektivno obiskujemo pričakovano pomembne regije — pravimo *presejanje*.

Regije na disku si lahko predstavljamo kot vozlišča v grafu, ki jih povezujejo reference. Na primer dve regiji A in B sta povezani takrat, ko regija B vsebuje kos datoteke, katere začetek se nahaja v regiji A. Na sliki 2 je prikazan primer takšnega grafa, kjer odebeljene povezave prikazujejo obhod, ki ga opravi program za slikanje diska s presejanjem. Vidimo, da obhod nikoli ne obišče regije `00000013`, saj se ne nahaja v isti povezani komponenti kot regije korenskega imenika. Prav tako ni obiskana regija `00000006`, kar pomeni, da je program na podlagi profila določil, da ta regija ni pomembna.



Slika 2: Disk predstavljen kot graf, kjer so vozlišča regije na disku povezave pa reference med njimi.

4. REZULTATI

V tem razdelku so predstavljeni rezultati testiranja metode presejanja, ki so jih pridobili avtorji članka [7]. Natančnost, časovni prihranek, kompatibilnost in celovitost so testirali s pomočjo obstoječih forenzičnih orodij in metod [9, 4, 5, 8]. Uporabljali so originalno sliko diska za kontrolo in presejano sliko, pridobljeno z uporabo lastne implementacije prototipa za presejanje.

Testiranje zgoraj omenjenih karakteristik je potekalo na dveh javno dostopnih slikah diskov, in sicer NIST CFReDS [2] in NPS nps-2009-domexusers [3].

Pohitritev je bila izmerjena tako, da so primerjali število prebranih sektorjev pri klasičnem slikanju diska in število prebranih sektorjev pri presejanju.

Pri natančnosti presejanja so želeli ugotoviti ali so klonirani deli diska po presenanju zares popolnoma identični originalnim. Ugotovili so, da so v vseh primerih klonirani deli diska popolnoma enaki originalnim na kontrolni sliki diska.

Celovitost so testirali tako, da so uporabili različna foren-

Tabela 1: Rezultati testiranja celovitosti

Primer	Profil	Pohitritev	Orodje	Celovitost
NIST CFReDS Hacking Case (4.6GB)	Registry	4.5x	log2timeline	100%
	Registry	4.5x	Regripper	100%
	IEHistory	5.0x	Pasco	100%
	Email	3.7x	Bulk_extractor	95%
	Registry	4.5x	Mactime	100%
NPS DOMEXUSERS (40GB)	Registry	13x	log2timeline	54%
	Registry	13x	Regripper	100%
	IEHistory	13x	Pasco	100%
	Registry	11x	Bulk_extractor	57%
	Registry	13x	Mactime	100%

zična orodja, vidna v tabeli 1, v stolpcu *Orodje*. Vsako izmed orodij je bilo uporabljeno na sliki originalnega diska, potem pa še na presejani sliki. Nato so primerjali rezultate, pridobljene s posameznim orodjem, na podlagi česar so lahko objektivno izmerili kolikšen del podatkov, ki jih orodje išče, je vsebovan v presejani sliki. Rezultati so vidni v tabeli 1. Faktor pohitritve presejanja je na disku CFReDS med 3.5 in 5.0, celovitost pa med 95% in 100%. Faktor pohitritve presejanja je na disku DOMEX USERS med 11.0 in 13.0, celovitost pa med 54% in 100%. Razlog za nizko celovitost pri testiranju nekaterih orodij na drugem disku, so avtorji pripisali temu, da je bilo veliko relevantnih podatkov shranjenih v sistemskih obnovitvenih točkah, katerih pa njihov prototip za presejanje ni zajel, ker profil ni bil ustrezno definiran. Iz tega sledi, da je v prihodnosti v podobne profile smiselno dodati tudi sistemske obnovitvene točke.

Presejanje so prav tako testirali na dveh forenzičnih primerih, in sicer na hipotetičnem primeru M57-Jean [1] in na realnem primeru, kjer je želel delodajalec ugotoviti, ali zaposleni kršijo pravilo o gledanju službeno neprimernih vsebin. Za oba primera so razvili poseben profil za presejanje. Nato so opravili forenzično preiskavo, pri kateri so uporabili presejanje. Pri prvem primeru so se ugotovitve preiskave popolnoma ujemale z referenčnimi ugotovitvami, torej je bila dosežena celovitost 100%. Faktor pohitritve v prvem primeru je bil 3.2. Pri drugem primeru so morali analizirati 3 različne diske velikosti 40 GB, 160 GB in 320 GB. Diska velikosti 40 GB jim s presejanjem ni uspelo zajeti, saj je bil poškodovan (kljub temu ga je bilo mogoče zajeti z orodjem FTK v3.2). Celovitost presejanja na disku velikosti 160 GB je bila več kot 99%, faktor pohitritve za ta disk je pa znašal 2.9. Avtorji so odkrili, da so datoteke, ki so manjkale po presejanju, vse v določeni veji zgodovine brskalnika Internet Explorer, česar v članku niso znali pojasniti. Celovitost presejanja na disku velikosti 320 GB je bila 100%, faktor pohitritve pa 9.6.

5. ZAKLJUČEK

Zaradi povečevanja diskov potrebujemo nove pristope za zajem forenzičnih dokazov. Predstavili smo pristop, ki združi dobre lastnosti kloniranja diskov in triaže. Rezultat je delna slika diska, ki je manjša in jo je mogoče hitreje ustvariti, analiziramo pa jo lahko z obstoječimi orodji. S pomočjo profilov lahko zajem slike avtomatiziramo in s tem izničimo potrebo po izvedencu na kraju zločina. Preizkus metode je pokazal velike pohitritve brez izgube pomembnega dokaznega gradiva. Preden se ta metoda lahko uveljavi v uporabi na resničnih primerih, bo potrebno še nekaj dela na dopolnjevanju profilov. Metoda temelji na pravilni izbiri profila in zato ne more nadomestiti kloniranja celotnega diska, saj so lahko ob napačni izbiri izgubljeni pomembni dokazi. Enako velja za triažna orodja, pri katerih pa dodatno izgubimo še preverljivost dokaznega gradiva. Opisana metoda po našem mnenju predstavlja dobro alternativo triažnim orodjem.

6. VIRI

[1] M57-jean forensic scenario. http://digitalcorpora.org/corpora/scenarios/m57-jean. Dostopano: 7. 5. 2016.

[2] Nist cfreds disk images. http://www.cfreds.nist.gov/dfr-test-images.html. Dostopano: 14. 5. 2016.

[3] Nps domexusers disk image. http://digitalcorpora.org/corp/nps/drives/nps-2009-domexusers/. Dostopano: 14. 5. 2016.

[4] J. R. Bradley and S. L. Garfinkel. Bulk extractor user manual. Technical report, 2013.

[5] B. Carrier. The sleuth kit and autopsy: forensics tools for linux and other unixes. http://www.sleuthkit.org, 2005. Dostopano: 10. 6. 2016.

[6] W. D. Garfinkel S, Nelson A and R. V. Using purpose-built functions and block hashes to enable small block and sub-file forensics. *Digital Investigation*, S13(23), 2010.

[7] J. Grier and G. G. Richard. Rapid forensic imaging of large disks with sifting collectors. *Digital Investigation*, 14:S34–S44, 2015.

[8] K. Gudjónsson. Mastering the super timeline with log2timeline. *SANS Institute*, 2010.

[9] K. Jones. Pasco v. 1.0. www.mcafee.com/us/downloads/free-tools/pasco.aspx. Dostopano: 10. 6. 2016.

[10] NIJ. New approaches to digital evidence processing and storage. *U.S. Department of Justice*, 2014.

[11] R. V. Richard GG. *Digital forensics tools: the next generation. Digital Crime and Forensic Science in Cyberspace*. Idea Group Publishing, 2006.

[12] R. G. Roussev V. Breaking the performance wall: the case for distributed digital forensics. In *Proceedings of the 2004 Digital Forensics Research Workshop*, 2004.

[13] B. A. Shaw A. A practical and robust approach to coping with large volumes of data submitted for digital forensic examination. *Digital Investigation*, 10, 2013.

Izrezovanje na podlagi izvlečkov podatkovnih sektorjev

[Seminarska naloga]

Tilen Faganel
Fakulteta za računalništvo in informatiko
Ljubljana, Slovenija
tf9957@student.uni-lj.si

Nina Habjan
Fakulteta za računalništvo in informatiko
Ljubljana, Slovenija
nh4160@student.uni-lj.si

Jan Kos
Fakulteta za računalništvo in informatiko
Ljubljana, Slovenija
jk1557@student.uni-lj.si

ABSTRACT

Izrezovanje na podlagi izvlečkov je tehnika za odkrivanje prisotnosti določenih datotek na podatkovnih nosilcih s primerjanjem izvlečkov posameznih podatkovnih sektorjev namesto celotnih datotek. Na ta način lahko identificiramo fragmentirane, nepopolne in celo delno spremenjene datoteke. Dosedanji poizkusi izrezovanja na podlagi izvlečkov so bili izvedeni na relativno majhnih množicah podatkov. Avtorji članka [7] izvedejo izrezovanje z bazo, ki obsega približno miljon ciljnih datotek, in pri tem odkrijejo nepričakovano visoko stopnjo napake zaradi ponavljajočih se podatkovnih struktur v dokumentih Microsoft Office in multimedijskih datotekah. V nadaljevanju predstavijo rešitev v obliki dveh algoritmov HASH-SETS in HASH-RUNS, ki omogočata odkrivanje prisotnosti datotek in njihovo rekonstrukcijo s pomočjo baze izvlečkov podatkovnih sektorjev. Pri demonstraciji tehnike si pomagajo z orodjem bulk_extractor, podatkovno bazo hashdb in implementacijo algoritma v jeziku Python. V seminarski nalogi si najprej pogledamo stanje na tem področju, osnovno idejo izrezovanja na podlagi izvlečkov podatkovnih sektorjev in nato še predlagano rešitev iz članka na primeru.

Keywords

izrezovanje, izvleček, forenzična preiskava, izrezovanje na podlagi izvlečkov, HASH-SETS, HASH-RUNS

1. UVOD

Na področju digitalne forenzike se izvlečki uporabljajo pogosto in za najrazličnejše namene. Eden izmed njih je tudi uporaba baz izvlečkov za iskanje znanih datotek, kar je običajna praksa tekom forenzične preiskave. Po zasegu podatkovnega nosilca se izračuna izvleček vsake datoteke, ki se ga nato primerja z izvlečki v podatkovni bazi. Če se izvlečka ujemata, gre za identični datoteki, kar potrjuje obstoj določene datoteke na zaseženem podatkovnem nosilcu. Tak način iskanja datotek je uspešen le pri datotekah, ki so popolne in niso bile spremenjene. Že malenkostna sprememba datoteke se namreč odraža v povsem drugačnem izvlečku.

Naslednja tehnika, ki se redno uporablja za iskanje datotek na zaseženih nosilcih, je tehnika izrezovanja. Le-ta omogoča odkrivanje datotek, ki so skrite znotraj drugih datotek. Gre za postopek iskanja datoteke na podlagi glave in noge, ki sta značilni za določen tip (format) datoteke. Ko sta znotraj večje datoteke locirani glava in noga, ki sta del manjše datoteke, je le-to mogoče izrezati iz večje datoteke in nato izračunati njen izvleček ter uporabiti prej opisani postopek za dokazovanje obstoja specifične datoteke na preiskovanem nosilcu podatkov.

Pristop izrezovanja na podlagi izvlečkov podatkovnih sektorjev, ki je predstavljen v članku [7], združuje prej opisana postopka odkrivanja datotek. Ideja tega pristopa je primerjanje izvlečkov posameznih podatkovnih sektorjev namesto celotnih datotek, kar omogoča tudi odkrivanje fragmentiranih, nepopolnih in celo delno spremenjenih datotek ter delov datotek, ki se nahajajo v datotekah swap navideznega pomnilnika.

Ker želijo kriminalci sporne datoteke, ki bi lahko privedle do njihove obsodbe skriti, izbrisati in spremeniti, ter se s tem izogniti odkritju, lahko opisana tehnika pomembno prispeva k odkrivanju dokazov, ki bi sicer ostali spregledani.

2. PREGLED PODROČJA

Kot smo že omenili, sta tako uporaba izvlečkov kot tehnika izrezovnja že uveljavljeni praksi digitalne forenzike. Idejo, ki združuje oba postopka - izrezovanje na podlagi izvlečkov pa je Garfinkel prvič predstavil kot del rešitve za izziv na delavnici Digital Forensics Research Conference (DFRWS) leta 2006 [4]. Iz surove datoteke je izluščil besedilo, na podlagi katerega je na spletu poiskal ciljne dokumente. Izračunal je izvlečke posemeznih blokov in jih ročno primerjal z izvlečki sektorjev surove datoteke. Na ta način je identificiral lokacijo datotek v izzivu. Tri leta kasneje je izdal orodje `frag_find`, ki avtomatizira omenjeni postopek.

Dandass je leta 2008 opravil analizo MD5 in SHA1 izvlečkov več kot 528 miljonov sektorjev iz 433 tisoč datotek. Kljub velikemu vzorcu datotek ni odkril bistvenih kolizij [2]. Leta 2009 je Collange v članku [1] raziskoval uporabo grafičnih procesorjev za pospešitev računanja izvlečkov in pri tem uvedel izraz izrezovanje na podlagi izvlečkov sektorjev (*hash-based carving*). Ni pa se ukvarjal z vprašanjem konkretne podatkovne baze, potrebne za poizvedovanje po izvlečkih,

in problemom ponavljajočih se blokov.

Leta 2013 je Key[8] razvil orodje `EnScript`, ki omogoča gradnjo slovarja izvlečkov iz seznama datotek, kakor tudi iskanje teh blokov na podatkovnih nosilcih. Orodje je, podobno kot `frag_find`, omejeno na iskanje relativno majhne množice datotek.

Garfinkel *et al.* [5] so za potrebe forenzičnih raziskav na datotekah ustvarili referenčni korpus, sestavljen iz približno milijona datotek, imenovan "GOVDOCS corpus". Gre za naključne datoteke, objavljene na spletni strani ameriške vlade. Poleg tega so ustvarili tudi scenarij treh povezanih zločinov v izmišljenem podjetju M57, ki zajema množico slik diskov, zajemov delovnega pomnilnika in omrežnega prometa. Oba korpusa datotek sta za testne primere uporabljena v članku [7] in nadaljnih raziskavah.

Leta 2010 se je Garfinkel [6] že ukvarjal z identifikacijo datotek na podatkovnih nosilcih na podlagi razločevalnih blokov vendar brez konkretnega algoritma. Fosterjeva [3] je kasneje analizirala 50 naključno izbranih blokov, ki se ponavljajo v različnih datotekah iz "GOVDOCS corpus" in ugotovljala razloge za njihovo ponavljanje.

V članku [10] so Young in ostali avtorji razširili idejo uporabe baze izvlečkov razločevalnih blokov poznanih datotek za zaznavanje prisotnosti le-teh na preiskovanem podatkovnem nosilcu. Članek zajema pregled osnovne ideje, možnosti uporabe različnih podatkovnih baz kot tudi opis osnovne implementacije baze izvlečkov `hashdb`.

Taguchi je v svojem delu [9] preučeval kompromise pri uporabi vzorcev različnih velikosti za naključno vzorčenje in računanje izvlečkov sektorjev pri triaži podatkovnih nosilcev. Ugotovil je, da so v različnih okoliščinah pri branju najbolj optimalni vzorci velikosti 64 KiB.

3. IZREZOVANJE NA PODLAGI IZVLEČKOV PODATKOVNIH SEKTORJEV

3.1 Ideja

Na vsako datoteko lahko gledamo kot na množico blokov, iz katerih je sestavljena (v članku [7] gre za bloke velikosti 4KiB). Ko se datoteka zapiše na podatkovni nosilec (npr. trdi disk), se posamezen blok zapiše na določen sektor diska. Sektor je najmanjša enota podatkovnega nosilca, ki je lahko dodeljena. Večina modernih podatkovnih nosilcev uporablja sektorje velikosti 4KiB, ki pa so znotraj operacijskega sistema predstavljeni kot osem 512B sektorjev, kar je bila običajna velikost sektorja v preteklosti.

Predlagana tehnika izrablja opisano strukturo datotek in način, na katerega se datoteke zapisujejo na podatkovne nosilce. Pristop temelji na izračunu izvlečkov za vsak posamezen blok datoteke. Ti izvlečki se nato shranijo v bazo izvlečkov, ki je kasneje uporabljena pri iskanju blokov teh datotek na podatkovnem nosilcu, ki je predmet forenzične preiskave. Na začetku preiskave se izračuna izvleček vsakega sektorja podatkovnega nosilca. Te izvlečke se nato primerja z izvlečki datotek, ki so bili predhodno izračunani in shranjeni v bazo. Ker gre za izvlečke posameznih blokov, nam to omogoča identificiranje fragmentov datotek, ki so lahko nastali kot posledica fragmentiranja ali z brisanjem in delnim prepisom izbrisane datoteke.

Na kratko je ideja postopka torej sledeča:

1. izračun izvlečkov posameznih blokov poznanih datotek
2. izračun izvlečkov posameznih sektorjev podatkovnega nosilca, ki ga preiskujemo
3. iskanje ujemanj

Pri tem je pomembno, da sta blok datoteke in sektor podatkovnega nosilca enake velikosti.

3.2 Postopek izrezovanja na podlagi izvlečkov podatkovnih sektorjev

Predlagani postopek sestavljajo štirje koraki, ki jih prikazuje slika 1:

1. **Gradnja podatkovne baze:** shranjevanje izvlečkov blokov znanih datotek v podatkovno bazo;
2. **Preiskovanje podatkovnega nosilca:** branje podatkovnega nosilca, ki ga preiskujemo in izračun izvlečka za vsak posamezni sektor ter primerjava teh izvlečkov z zapisi v podatkovni bazi;
3. **Izbira kandidatov:** Oblikovanje množice znanih datotek, ki se glede na primerjane izvlečke verjetno nahajajo na preiskovanem podatkovnem nosilcu. Določen izvleček sektorja se namreč lahko ujema z izvlečkom, ki pripada eni ali več znanim datotekam;
4. **Sestavljanje ciljne datoteke:** iskanje zaporedja ujemajočih se blokov za vsakega identificiranega kandidata na preiskovanem podatkovnem nosilcu in preslikava le-teh na ustrezno znano datoteko.

Do ujemanja izvlečka bloka znane datoteke in izvlečka sektorja preiskovanega podatkovnega nosilca pride v naslednjih primerih:

- na podatkovnem nosilcu se nahaja nespremenjena kopija znane datoteke - ne glede na to ali se nahaja v dodeljenem ali nedodeljenem prostoru ali je označena kot izbrisana;
- na podatkovnem nosilcu se je nekoč nahajala kopija znane datoteke, ki je bila nato izbrisana in delno prepisana. Na podatkovnem nosilcu se nahajajo fragmenti, ki so ostanek znane datoteke;
- na podatkovnem nosilcu se nahaja datoteka, ki ima nekatere sektorje enake kot znana datoteka. Postopek v tem primeru odkrije sektorje, ki so skupni obema datotekama;
- znana datoteka je vdelana v večjo datoteko na podatkovnem nosilcu in poravnana na mejo sektorja.

Idealni algoritem bi vse te primere obravnaval hkrati.

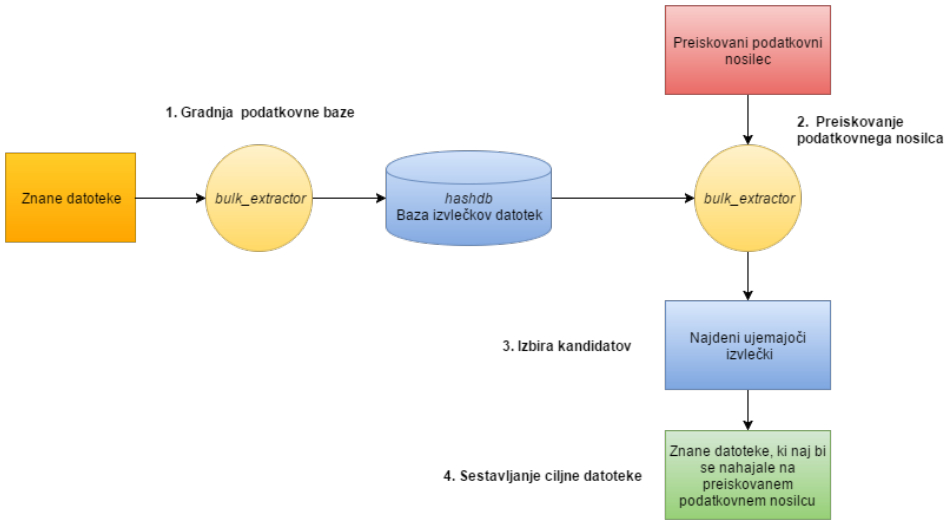


Figure 1: Postopek izrezovanja na podlagi izvlečkov podatkovnih sektorjev

```
8102 0000 8202 0000 8302 0000 8402 0000
8502 0000 8602 0000 8702 0000 8802 0000
8902 0000 8a02 0000 8b02 0000 8c02 0000
8d02 0000 8e02 0000 8f02 0000 9002 0000
```

Figure 2: Pogost blok, ki je del dokumentov Microsoft Office

3.3 Težave

3.3.1 *Pogosti bloki*

Do nezanemarljivih težav pride, ker se lahko posamezen blok pojavi v več kot eni izmed znanih datotek. Fosterjeva [3] take bloke imenuje "pogosti bloki" (*common blocks*). Med njimi se največkrat pojavi blok samih NULL vrednosti, ki se uporablja za inicializacijo praznih medijev ter se nahaja tudi v mnogih dokumentih. Tak blok mora biti torej obravnavan na poseben način, saj bi bil sicer algoritem neučinkovit, prišlo pa bi lahko tudi do izčrpanja pomnilnika.

Naslednji "pogost blok", ki ga opisuje Fosterjeva je blok monotono naraščajočih 32-bitnih števil, ki ga prikazuje slika 2.Tak blok se nahaja v vseh dokumentih Microsoft Office, ki vsebujejo vgrajene datoteke (npr. slike). Ta vzorec pripada tabeli dodeljenih sektorjev (*SAT - Sector Allocation Table*). Začetna vrednost je odvisna od lokacije vgrajene datoteke, zaradi česar je verjetnost ujemanja struktur SAT dveh poljubnih datotek Microsoft Office majhna, hkrati pa je velika verjetnost, da bo prišlo do ujemanj, če primerjamo dve veliki zbirki takih dokumentov.

Pojavljanje pogostih blokov predstavlja težavo z dveh vidikov:

- ker se taki bloki pojavljajo v več datotekah, jih ne moremo uporabiti v koraku izbire kandidatov - blok, ki se pojavi v 100 različnih znanih datotekah ne more dokazovati, da je katerakoli od teh datotek prisotna na preiskovanem podatkovnem nosilcu;
- večja kot je podatkovna baza, več pogostih blokov bomo odkrili.

Potreben je pristop, s katerim pogoste bloke prepoznamo in ignoriramo še preden pride do kolizije, saj je nemogoče, da bi vse poznali že vnaprej.

3.3.2 *Velikost sektorjev in poravnava*

Kot že omenjeno, se v članku [7] uporabljajo bloki velikosti 4KiB. Pri podatkovnih nosilcih, katerih velikost sektorja je 512B, je zato potrebno združevanje osmih sektorjev v "supersektor", nad katerim je nato izračunan izvleček.

Prednost večjih sektorjev je manjše število izvlečkov, kar omogoča, da celotno bazo izvlečkov hranimo v RAM-u, s čimer zagotovimo hiter dostop.

Težava se pojavi pri poravnavi z datotečnim sistemom. Sektorji morajo biti namreč poravnani z bloki datotečnega sistema, če želimo, da se izvlečki ujemajo. Poravnavo dosežemo s poravnavanjem sektorjev z začetkom datotečnega sistema. V nekaterih primerih začetka datotečnega sistema ne moremo določiti - bodisi zaradi okvarjene tabele razdelkov bodisi če je na podatkovnem nosilcu obstajal datotečni sistem, ki se je začel na drugem mestu.

Težavo rešimo z uporabo drsečega okna velikosti 4KiB, ki ga premikamo po 512B čez celoten podatkovni nosilec in računamo izvlečke. Tako dobimo osem množic izvlečkov - vsaka z drugačno pozicijo začetnega sektorja po modulu 8. Zdi se, da nam to prinese toliko dodatnega dela, kot če bi za velikost sektorja izbrali 512B. Vendar pa je tako preiskovanje potrebno le v primeru podatkovnih nosilcev, kjer je velikost sektorja 512B in kjer preiskovalec nima podatka o začetku datotečnega sistema oz. o predhodnih datotečnih sistemih na tem nosilcu. Prav tako je na tak način potrebno preiskati

le podatkovni nosilec, medtem ko v koraku gradnje podatkovne baze to ni potrebno, zato je tudi sama podatkovna baza osemkrat manjša, kot bi bila sicer.

4. TESTNI PRIMER

Za testiranje algoritmov izrezovanja na podlagi izvlečkov je bilo uporabljeno fiktivno kaznivo dejanje uslužbenca Jo-ja, ki zbira slike mačk. Dejanje je del podatkovne baze "M57-Patents", medtem ko so podatki - 82 JPEG datotek, 2 QuickTime datoteki in 4 MPEG4 datoteke - del podatkovne baze "Monterey Kitty", zbrane v mestu Monterey CA. Ti podatki so bili nato uporabljeni kot nadomestilo za otroško pornografijo. Na tem primeru bomo predstavili potek raziskave na sliki medija iz službenega računalnika, ki ga je imel v lasti Jo.

Na omenjenih datotekah je bilo opravljeno zbiranje izvlečkov podatkovnih sektorjev s pomočjo odprtokodnega programa `bulk_extractor`. Ti izvlečki so bili nato shranjeni v posebno podatkovno bazo imenovano `hashdb`, ki je posebej namenjena shranjevanju in primerjanju izvlečkov blokov. Omenjeni orodji nato uporabimo za izvedbo prvih dveh korakov izrezovanja na podlagi izvlečkov. V nadaljevanju je po korakih predstavljeno izrezovanje na tem primeru.

4.1 Gradnja podatkovne baze

Prvi potreben korak za uspešno analizo je gradnja podatkovne baze izvlečkov vseh sektorjev znanih datotek, na tak način, da jih bomo lahko kasneje učinkovito primerjali z medijem, ki ga analiziramo. Z uporabo programa `bulk_extractor` je bila ustvarjena podatkovna baza `hashdb`, ki je vsebovala izvlečke vseh 4KiB blokov znanih datotek. Po opravljeni gradnji je podatkovna baza vsebovala 50206 izvlečkov, pridobljenih iz 88 različnih datotek. Izdelava histograma nad podatkovno bazo je pokazala, da so vsi izvlečki iz medija unikatni, kar pomeni, da nista nobeni dve datoteki enaki.

Za učinkovito analizo je bila dodana še ena večja podatkovna baza izvlečkov datotek, ki je pripomogla pri primerjavi oz. analizi. V tem primeru so bile izbrane datoteke iz "GOV-DOCS corpus", na katerih so s pomočjo orodja `bulk_extractor` zgradili `hashdb` podatkovno bazo izvlečkov sektorjev datotek, ki je štela 119.687.300 izvlečkov, pridobljenih iz 909.815 datotek. Izmed teh se je 117.213.026 izvlečkov pojavilo samo enkrat, 514.238 dvakrat, 60.317 trikrat itd. Zanimiv podatek je bil, da je bil eden od izvlečkov v podatkovni bazi prisoten kar 11.434-krat. Zgrajena podatkovna baza je bila nato združena s prvotno, ta pa je nato tvorila osnovo za kasnejšo analizo.

4.2 Preiskovanje podatkovnega nosilca

Po uspešni gradnji podatkovne baze izvlečkov znanih datotek nastopi korak preiskovanja in analiza podatkovnega nosilca. Za iskanje 13GiB velike slike medija je bila uporabljena združena podatkovna baza in orodje `bulk_extractor`. Slednje orodje razbije sliko v strani, velike 16MiB, katere nato pregleda ali so prazne ali ne. V primeru, da je stran prazna, jo preskoči. Vsaka stran je nato naknadno razbita na 32.768 prekrivajočih se blokov velikosti 4KiB. Za vsak blok je nato, s pomočjo algoritma MD5, izračunan izvleček, ki je nato uporabljen kot poizvedba v podatkovno bazo. Byte-i na koncu strani so pri procesu združeni z byte-i na začetku naslednje strani. V analizi je bilo zajetih skupno 394 strani (6,3 GiB), kar predstavlja približno 12,9 milijonov izvlečkov sektorjev. Preiskovanje je v osnovi večnitno (tako izračun izvlečkov, kot nato poizvedbe v podatkovni bazi), s čimer lahko zagotovimo znatne pohitritve na večjedrnih sistemih. Kljub temu pa smo pri hitrosti oz. učinkovitosti še vedno omejeni s hitrostjo I/O sistemov in ne procesorja.

Po opravljeni analizi so bili rezultati na voljo v obliki tekstovne datoteke. Primer take datoteke lahko vidimo v odseku 3. Ta je vsebovala vse izvlečke sektorjev, za katere je bilo najdeno ujemanje. Najdenih je bilo 33.847 takih ujemanj. Za vsako ujemanje lahko vidimo, v koliko datotekah se je izvleček pojavil. To informacijo nam poda "count" stolpec v rezultatih. V drugem stolpcu lahko vidimo izvleček sektorja, ki je bil najden. V prvem stolpcu nato najdemo lokacijo sektorja na mediju. Iz primera lahko vidimo, da se je isti sektor pojavil na treh lokacijah, ki so 512B narazen. Če pregledamo disk, lahko vidimo, da se na teh lokacijah pojavi vzorec "ffff ff00", ki se nato nekajkrat ponovi. V tretjem stolpcu se nahajajo dodatne informacije o sektorju, npr. št. ponovitev sektorja (39 za prvo vrstico) in polje "flags", ki nam pove, ali se blok ujema s katerim od pravil za identifikacijo pogostih blokov. Pravila so bolj podrobno opisana v poglavju 4.3. Najdeni izvlečki se nato uporabijo v naslednjih dveh fazah, sprva za identifikacijo ciljnih datotek, ki so lahko prisotne na mediju, in nato pri poskusu rekonstrukcije teh datotek.

4.3 Izbira kandidatov

V prvotni verziji algoritma za izrezovanje na podlagi izvlečkov sektorjev ni bilo eksplicitnega koraka za izbiro kandidatov. Tako smo od preiskovanja podatkovnega medija takoj prešli na rekonstrukcijo datotek, in sicer tako, da smo poskusili rekonstruirati vsako datoteko, za katero je bil najden sektor, ki se je ujemal. Tak pristop se je izkazal za problematičnega, saj so nekateri najdeni sektorji lahko prisotni v več tisoč različnih datotekah. Zato je prišlo do tega, da je bilo potrebno rekonstruirati zelo veliko število datotek, kar pa je zelo poslabšalo performance in obremenilo analitika. To je privedlo do neefektivne rekonstrukcije datotek. Kot možna rešitev je bil dodan vmesen korak izbire kandidatov. Sprva je bila selekcija omejena na najdene sektorje, ki so bili v podatkovni bazi prisotni samo enkrat. To je sicer občutno zmanjšalo število nepotrebnih datotek, ki jih je bilo potrebno rekonstruirati, vendar je bilo vseeno prisotnih nekaj 100 odvečnih datotek. Kljub temu da se je v podatkovni bazi sektor pojavil samo enkrat, se je pogosto zgodilo, da se je sektor pojavil v več različnih datotekah. Pogosto so bile to binarne strukture datotek, ki jih proizvedeta Microsoft Office in Adobe Acrobat. Težava se pojavi tudi v primeru, ko je normalno, da se blok v podatkovni bazi pojavi večkrat. Npr. če imamo dve isti verziji video datoteke, le da je ena nekoliko skrajšana. V tem primeru bo ena verzija datoteke ignorirana.

Izbira kandidatov je bila razširjena tako, da se upošteva frekvenca pojavitev sektorja v podatkovni bazi in karaktaristike vzorca byte-ov iz katerih je sektor sestavljen. Omenjena pravila se upoštevajo že pri preiskovanju podatkovnega nosilca, kjer so najdeni sektorji ustrezno označeni.

- *Test s pomočjo rampe* - Najpogostejše nepomembne

```
86435328      736d99610d0097be78651ecdae4714bb     {"count":39,"flags":"H"}
86435840      736d99610d0097be78651ecdae4714bb     {"count":39,"flags":"H"}
86436352      736d99610d0097be78651ecdae4714bb     {"count":39,"flags":"H"}
1231920640    90ccbdf24a74c8c05b94032b4ce1825d     {"count":1,"flags":"H"}
1231924736    9403e1cac89e860b93570ac452d232a5     {"count":1}
1231928832    b59246507f2bedb21957fae92bcf37d0     {"count":1}
1351669248    1e79c17035c597269b6fedf614663a1e     {"count":2,"flags":"HW"}
```

Figure 3: Primer formata rezultatov analize izvlečkov sektorjev. Prvi stolpec je lokacija najdenega sektorja na mediju, drugi stolpec je MD5 izvleček sektorja in tretji stolpec vsebuje morebitne dodatne metapodatke o najdenem sektorju.

datoteke so tiste, ki vsebujejo alokacijske tabele Microsoft Office (ali podobne programske opreme). Take datoteke so pogosto podobne velikosti in vsebujejo enake začetne vrednosti. Na sistemu tako tipično najdemo več deset tisoč takih sektorjev. Take bloke lahko s preprostim testom, ki preverja, če se vsaj pol byte-ov ujema z vzorcem omenjenih datotek, avtomatsko izločimo. Tak način se je izkazal za zadovoljiv.

- *Test belih znakov* - Naslednji pogosti tip sektorja, ki ga zasledimo, vsebuje približno 100 presledkov, ki so zaključeni z znakom za novo vrstico. Tak tip blokov je pogost pri datotekah JPEG, ki jih ustvari programsko orodje Adobe Photoshop. Če preiskujemo medij, ki vsebuje veliko število slik, lahko vsebuje veliko št. takih blokov. Take bloke odstranimo, če vsebujejo več kot tri četrtine belih znakov.

- *4-byte histogram test* - Pogosta binarna struktura, ki se pojavi, je vzorec vrednosti 4 byte-ov, ki se bodisi ponavlja bodisi alternira. Vzorci se najpogosteje pojavijo v datotekah tipa Apple QuickTime in Microsoft Office. Take vzorce nato s pomočjo histograma, ki ga izračunamo na sektorju, zaznamo in izločimo. Malo verjetno je, da bo to pravilo izločilo kakšno besedilo oz. sliko, saj take datoteke navadno ne vsebujejo dolgih nizov 4 byte-nih vzorcev.

- *Test entropije* - V [7] lahko zasledimo, da imajo pogosti sektorji ponavadi nizko entropijo. To informacijo lahko uporabimo tako, da izločimo sektorje, ki vsebujejo dovolj nizko entropijo. Za vsak sektor tako izračunamo Shannonovo entropijo, katero nato primerjamo z nastavljeno mejo. Če je entropija pod to mejo, sektor zavržemo.

Vsa omenjena pravila so bila uporabljena na 677 razločevalnih blokih, pridobljenih iz zaseženega medija, ki so imeli ujemanje v bazi izvlečkov zgrajeni iz referenčne množice datotek "GOVDOCS corpus" in niso v štirih najdenih datotekah. Prvo pravilo je ujelo 200 blokov, pravilo s histogrami pa 400. Test z entropijo je pri meji 7 ujel 600 blokov. Ti so bili identični kot kombinacija ujemanj prvega in tretjega pravila. Tako lahko vidimo, da so tako pravila 1 - 3, kot tudi Shannonova entropija, uspešna pri izločanju nepomembnih blokov. Tudi v primeru, ko omenjena pravila označijo kakšen blok kot nepomemben, v resnici pa temu ni tako, to ne predstavlja večjih težav, saj izrezovanje tolerira izgubo nekaj blokov. Dokler obdržimo večino blokov v datoteki, nam manjše število lažno pozitivnih primerov ne škodi.

Izbira kandidatov je implementirana na sledeč način. Najprej se uporabi funkcionalnosti podatkovne baze `hashdb`, da pridobimo podatke o datotekah, katerim najdeni bloki pripadajo. Seznam nato podvojimo in za vsak sektor ugotovimo, v koliko datotekah se nahaja. Če se sektor nahaja v manj kot N datotekah (privzeto 20), potem so te datoteke dodane v seznam kandidatov. Nato se za vsak ustrezen blok zapišejo še dodatni metapodatki. Na koncu se za vsak blok poženejo še opisana pravila. V primeru, da ga nobeno pravilo ne odstrani, se blok doda na seznam kandidatov.

4.4 Sestavljanje ciljne datoteke

Po opravljeni izbiri kandidatov se izvlečki blokov, ki pripadajo kandidatom, grupirajo glede na datoteko, v kateri se nahajajo. Nato s pomočjo naslednjih algoritmov datoteke rekonstruiramo:

- `HASH-SETS` - detekcija odstotka ciljne datoteke.
- `HASH-RUNS` - lociranje ciljnih datotek.

4.4.1 HASH-SETS

`HASH-SETS` je preprost, prostorsko učinkovit algoritem, ki s pomočjo izvlečkov blokov ugotovi, kolikšen odstotek blokov pripada vsaki datoteki, ki je prisotna na mediju.

Algoritem ne hrani točne lokacije vsakega bloka na mediju, kar naredi algoritem zelo efektiven. Implementacija je enostavna:

1. Seznam kandidatov je ustvarjen s pomočjo algoritma izbire kandidatov, ki je opisan v 4.3.
2. Za vsak blok H, ki se nahaja v seznamu virov kandidatov:
   (a) Za vsako ciljno datoteko T, ki vsebuje blok H:
      i. Če je T kandidat, dodaj 1 v oceno datoteke.
3. Za vsako ciljno datoteko T izračunaj odstotek, ki pove, kolikšna je prisotnost datoteke, tako, da deliš oceno datoteke s številom blokov v ciljni datoteki T.
4. Ciljne datoteke so uredi po velikosti glede na izračunan odstotek prisotnosti.
5. Če odstotek presega predefinirano mejo, jo izpiši za namen kasnejšega procesiranja.

Ime ciljne datoteke	Ocena	Začetni sektor	Začetni blok	Končni blok	Sektor (mod 8)	Rekonstruirani odstotek	Alocirana datoteka na mediju
...							
MontereyKittyHQ.m4v	6132	18639703	0	6131	7	100%	.../MontereyKittyHQ.m4v
TiggerTheCat.m4v	3059	3532519	0	3058	7	100%	.../TiggerTheCat.m4v
KittyMaterial/Cat.mov	1374	18696759	0	1393	7	100%	.../Cat.mov
466982.csv	4	2932551	0	3	7	0%	.../Cache/F5433139d01
DSC00072.JPG	234	14306831	0	233	7	100%	.../DSC00072.JPG
...							

Table 1: Izvleček rezultatov analize medija.

Prvotna verzija omenjenega algoritma je za oceno uporabljala inverzno dokumentno frekvenco $\frac{1}{N}$, vendar je bilo to kasneje spremenjeno na preprosto štetje števila blokov v datoteki. Tako je ocena izključno funkcija ciljne datoteke in medija ter ne konstrukcije podatkovne baze izvlečkov.

4.4.2 HASH-RUNS

Algoritem `HASH-SETS` zazna prisotnost ciljnih datotek, vendar ne pozna njihovih lokacij na mediju, saj je podatek o lokaciji zavržen. Algoritem `HASH-RUNS` poleg poročanja lokacije vsebuje še sledeče dodatne izboljšave:

- ustrezno zna ravnati s ciljnimi datotekami, ki se na mediju pojavijo na več lokacijah;
- upošteva in izkorišča dejstvo, da se soležni logični sektorji v datoteki pogosto nahajajo na soležnih fizičnih sektorjih medija;
- upošteva dejstvo, da imajo različni bloki v datoteki enako (mod 8) vrednost;
- zazna in združi sekvence znanih blokov, ki so ločeni z NULL bloki.

Algoritem začne s podatkovnimi strukturami, ki jih ustvari `HASH-SETS` algoritem. Nato identificira vse sekvence blokov na fizičnem mediju, ki pripadajo logični sekvenci ciljnih datotek. Ti bloki so nato urejeni glede na zaporedno številko logičnega bloka v ciljni datoteki in se izpišejo analitiku.

4.5 Rezultati algoritma

Zaseženi medij (opisan v 4) je bil uspešno analiziran z uporabo celotnega opisanega postopka. Primer formata rezultatov je prikazan v tabeli 1. Algoritem `HASH-RUNS` je poleg odstotka prisotnih blokov uspešno pridobil še pomembne dodatne informacije:

- algoritem `HASH-RUNS` je določil, da je bila datoteka "Cat.mov" v celoti prisotna na mediju, saj so bile rekonstruirane sekvence ločene s praznimi bloki in so zato lahko bile združene;
- v nasprotju z algoritmom `HASH-SETS`, je algoritem `HASH-RUNS` dodatno detektiral več primerov, ko je bilo na mediju prisotnih več kopij iste ciljne datoteke;
- z uporabo algoritma `HASH-RUNS` se je število lažno pozitivnih ujemanj občutno zmanjšalo. Algoritem `HASH-SETS` je identificiral 46 možnih ujemanj s podatkovno bazo "GOVDOCS corpus", medtem ko je algoritem `HASH-RUNS` identificiral 4 pravilna ujemanja in samo 4 nepravilna ujemanja zaradi pogostih blokov. Večja natančnost izhaja iz dejstva, da se rekonstruirane sekvence sektorjev in logična številka blokov povečujejo po korakih in ustreznem intervalu;
- vse napačno označene datoteke so imele oceno 3 ali manj in različno vrednost sektorjev od rekonstruiranih datotek. Vse datoteke znotraj enega datotečnega sistema morajo namreč imeti isto vrednost.

5. ZAKLJUčEK

Spoznali smo osnovno idejo izrezovanja na podlagi izvlečkov podatkovnih setorjev in si na primeru ogledali predlagano rešitev iz članka [7], ki uporablja orodje `bulk_extractor` za gradnjo baze izvlečkov blokov znanih datotek ter izračun izvlečkov sektorjev podatkovnega nosilca. Rešitev uporablja bazo `hashdb` za hitro poizvedovanje in začetno iskanje korelacij. Kljub relativno visokemu interesu za uporabo te tehnike, je v članku predstavljen eden prvih algoritmov in njegova referenčna implementacija, ki lahko obravnava večjo množico podatkov poznanih datotek. V nasprotju z dosedanjimi dognanji je bilo ugotovljeno, da prisotnost blokov z visoko entropijo ne pomeni nujno tudi prisotnosti znane datoteke na nosilcu. Pri večjih množicah podatkov se namreč pojavlja precej blokov, ki imajo visoko entropijo, vendar niso razločevalni. Nadaljni razvoj tovrstnih algoritmov lahko pomembno prispeva k odkrivanju dokazov, ki bi sicer ostali spregledani.

Tehnika je sicer neodvisna od uporabljenega datotečnega sistema, moramo pa se zavedati tudi njenih omejitev. V primeru, da podatkovni nosilec vsebuje kriptiran datotečni sistem, ga je potrebno pred izvedbo postopka dešifrirati, da lahko dostopamo do nešifriranih podatkovnih blokov.

Učinkovitost algoritma bi lahko izboljšali z vnaprejšnim označevanjem pogostih blokov že v fazi grdanje baze izvlečkov. Na ta način bi se izognili testiranju razločevalnih blokov ob vsaki pojavitvi istega sektorja na podatkovnem nosilcu. Poleg tega bi z izvedbo klasifikacije v tej fazi lahko uporabili zahtevnejše in posledično tudi boljše teste, saj hitrost gradnje baze ni tako kritična kot hitrost končne analize nosilca. Hkrati bi se lahko izognili shranjevanju datotek, ki so v celoti sestavljene iz pogostih blokov, saj jih z omenjenim postopkom ni mogoče rekonstruirati. Pri analizi bi lahko upoštevali še alokacijski status posameznih datotek, saj lahko pričakujemo, da so vsi bloki dolečene datoteke bodisi alocirani ali nealocirani, vendar bi to zahtevalo dodaten in računsko zahteven korak filtriranja.

6. REFERENCES

[1] S. Collange, Y. S. Dandass, M. Daumas, and D. Defour. Using graphics processors for parallelizing hash-based data carving. *CoRR*, abs/0901.1307, 2009.

[2] Y. S. Dandass, N. J. Necaise, and S. R. Thomas. An empirical analysis of disk sector hashes for data carving. *J. Digit. Forensic Pract.*, 2(2):95–104, Apr. 2008.

[3] K. Foster. *Using distinct sectors in media sampling and full media analysis to detect presence of documents from a corpus*. PhD thesis, Monterey, California. Naval Postgraduate School, 2012.

[4] S. Garfinkel. Dfrws 2006 challenge report, 2006. [dostopano 10.5.2016].

[5] S. Garfinkel, P. Farrell, V. Roussev, and G. Dinolt. Bringing science to digital forensics with standardized forensic corpora. *Digital Investigation*, 6, Supplement:S2 – S11, 2009. The Proceedings of the Ninth Annual {DFRWS} Conference.

[6] S. Garfinkel, A. Nelson, D. White, and V. Roussev. Using purpose-built functions and block hashes to enable small block and sub-file forensics. *Digit. Investig.*, 7:S13–S23, Aug. 2010.

[7] S. L. Garfinkel and M. McCarrin. Hash-based carving: Searching media for complete files and file fragments with sector hashing and hashdb. *Digital Investigation*, 14, Supplement 1:S95 – S105, 2015. The Proceedings of the Fifteenth Annual {DFRWS} Conference.

[8] K. Simon. File block hash map analysis, 2013.

[9] J. K. Taguchi. Optimal sector sampling for drive triage, 2013.

[10] J. Young, K. Foster, S. Garfinkel, and K. Fairbanks. Distinct sector hashes for target file detection. *Computer*, 45(12):28–35, Dec 2012.

Forenzična analiza podatkovne baze skozi pregled notranjih podatkovnih struktur

Janko Purgaj
Fakulteta za računalništvo in informatiko
Večna pot 113
Ljubljana, Slovenija
jp9237@student.uni-lj.si

Andrej Habazin
Fakulteta za računalništvo in informatiko
Večna pot 113
Ljubljana, Slovenija
ah8923@student.uni-lj.si

ABSTRACT

Dandanes je večina podatkov, s katerimi imamo opravka, shranjena v digitalni obliki. Forenzična analiza je tako največkrat osredotočena na restavriranje digitalnih vsebin in rekonstrukcijo dejanj uporabnika iz samih posnetkov sistema, na katerem je uporabnik izvajal akcije.

Predmet raziskave v tem delu so tako programske tehnike restavriranja podatkov iz relacijskih podatkovnih baz (ang. DMBS). Pri tem je bila uporabljena tehnika klesanja podatkov iz datotek [1]. Praviloma moramo s to tehniko uporabiti spremljajoče metapodatke da pridobimo vsebino datoteke.

V prvem delu bomo tako spoznali kateri parametri so pomembni z vidika analize in zajema podatkov in kako ustvariti orodje, ki zna prebrati podatke iz podatkovne baze, ne glede na verzijo in tip produkta.

V drugem delu nato z eksperimentalnim delom analiziramo delovanje orodja za rekonstrukcijo podatkov. Skozi različne eksperimente preverimo ali je rekonstrukcija podatkov pogojena z verzijo operacijskega sistema, koliko podatkov lahko rešimo iz okvarjenih podatkovnih tabel. Nenazadnje spoznamo tudi, koliko časa in kje se je na voljo podatek po tem, ko je iz podatkovne baze izbrisan.

Keywords

Podatkovne baze, klesanje, dekonstrukcija, digitalna forenzika

1. UVOD

Za organizirane zbirke podatkov (kar podatkovne baze nedvomno so), velja da so podatki predstavljeni strukturirano, organizirani v več različnih datotek in različnih podatkovnih formatih. Prav tako je struktura podatkov različna od produkta do produkta, celo med posameznimi generacijami se način zapisovanja in struktura podatkov precej spreminja. Ravno zaradi tega dejstva klasična tehnika klesanja podatkov ne zadostuje, saj v najboljšem primeru utegnemo pridobiti zgolj drobec podatkov, ki nas zanima. V podatkovnih bazah veliko vlogo igrajo tudi meta podatki (katalog), brez katerih podatkovna baza ne bi mogla delovati.

Tehnike, ki omogočajo forenzično analizo vsebine podatkovne baze so:

- Definiranje parametrov splošnega formata za prebiranje podatkov različnih relacijskih podatkovnih baz.
- Primerjava različnih načinov shranjevanja podatkov v podatkovnih bazah in njihov vpliv na forenzično analizo.
- Predstavitev orodja, ki z vzvratnim inženiringom določi parametre shranjevanja podatkovne baze s pomočjo iterativnega nalaganja sintetičnih podatkov, izvajanja SQL poizvedb in primerjave sprememb na datotečnem nivoju.
- Predstavitev orodja, ki iz podane slike diska ali posnetka vsebine delovnega pomnilnika lahko:
  - Poišče strani, v katerih se nahajajo podatki relacijskih podatkovnih baz, celo za različne produkte in možne konfiguracije.
  - Restavrira logično shemo (podatkovne tabele in omejitve na njih), kot tudi podatke, predstavljene v tabelarični obliki
  - Izvleček različnih spremenljivih podatkov, kot na primer: število brisanih vrstic ali vrednosti, ki bodo zapisane namesto obstoječih
  - Zaznavanje dokazov o uporabniških dejanjih, kot na primer: vrstni red dodajanja podatkov, katere vrstice je uporabnik prebral, itd.

Slika 1 prikazuje arhitekturo rešitve. V poglavju 2 bomo predstavili osnove shranjevanja podatkov v straneh relacijske podatkovne baze in definicijo parametrov, ki so pomembni z vidika razčlenjevanja in restavriranja podatkov iz strani. Prav tako bodo v tem poglavju tudi izpostavljene pomembne podatkovne strukture z vidika shranjevanja podatkov v bazi in osnove začasnega/spremenljivega (volatile) shranjevanja in posodabljanja podatkov.

V poglavju 3 bodo predstavljeni zanimivi vidiki in kompromisi, ki se jih podatkovne baze morajo posluževati za učinkovito delovanje. Ti parametri so namreč zelo pomembni

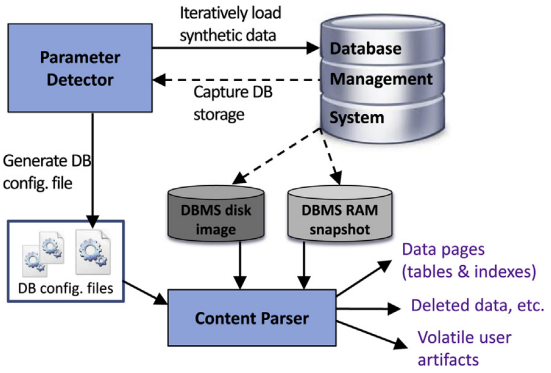


Figure 1: Visokonivojski pregled parametrov in arhitektura rešitve

pri odkrivanju zaporedja akcij, ki jih je uporabnik izvedel nad podatkovno bazo. Poglavje 4 o eksperimentalnem delu pokriva analizo rezultatov, ki so jih raziskovalci pridobili za različne podatkovne baze in njihove konfiguracije. Delo se zaključi s povzetkom vseh sorodnih raziskav, v samem zaključku so navedene tudi zanimive iztočnice za prihodnje obetavne raziskave.

2. STRUKTURA PODATKOVNE BAZE

Podatki so v relacijskih podatkovnih bazah tipično razdeljeni v strukture podatkov, imenovane strani, ki so fiksne velikosti med 4-8 KB. Razlog za tovrsten pristop je v tem, da fiksna velikost strani olajša shranjevanje podatkov, tudi mehanizmi za zagotavljanje predpomnjena so bistveno manj zapleteni zaradi tega. Velikost strani je sicer možno spreminjati, sprememba zahteva ponovno izgradnjo vseh podatkovnih struktur v podatkovni bazi.

Velikosti strani prav tako ni možno nastavljati za posamezne podatkovne tabele, v najboljšem primeru jo je možnost nastavljati za celoten logični prostor tabel (eng. tablespace). Pri shranjevanju podatkov v podatkovnih bazah imamo opravka z dvema tipoma metapodatkov: Splošne informacije o tem kje in kako so podatki shranjeni v tabelah in metapodatki o vsebini strani podatkovne baze. Izziv, ki ga za forenzično analizo predstavlja, je restavriranje vsebine podatkovne baze iz slike diska zgolj z informacijami, ki jih najdemo v metapodatkih strani in strani samih.

Na sliki 2 je shema strani, ki ima vseh relacijskih podatkovnih bazah enako strukturo, ki jo sestavljajo: zaglavje (header), imenik vrstice (row directory) in podatki vrstice. V zaglavju strani so shranjeni splošni podatki o strani (ali gre za tabelo, indeks?, komu stran pripada?) in se tipično nahaja na začetku strani. Imenik vrstice shranjuje pozicijo vrstice, ko se vrstice v tabeli dodajajo ali odstranjujejo. Struktura se lahko nahaja bodisi na koncu strani ali neposredno za zaglavjem. Podatki vrstice shranjujejo vsebine vrstice podatkovne tabele, skupaj z nekaterimi dodatnimi podatki.

2.1 Struktura strani podrobneje

Vsebino vsake komponente na strani podatkovne baze je možno opisati s splošnim naborom parametrom. Načeloma bi lahko razvili orodje, specifično za vsak tip podatkovne baze, vendar s tem izgubimo na splošnosti forenzičnega pristopa k preiskave poljubne podatkovne baze.

Veliko bolje je določiti splošne parametre, ki so pomembni z vidika forenzične preiskave. Za vsak tip podatkovne baze je potem potrebno zgolj spisati logiko, ki za dano strukturo prebere željeni parameter. Prednosti tega pristopa so, da lahko hitro in neodvisno od arhitekture, razvijemo uporabno diagnostiko, ki nam omogoča členitev in branje podatkov iz strani podatkovne baze.

2.1.1 *Parametri zaglavja strani*

Preden pričnemo z razlago, je potrebno omeniti, da sami parametri niso shranjeni zvezno znotraj zaglavja, medtem ko njihovi naslovi so. Zaglavje vsebuje splošni identifikator strani (General Page Identifier). Naslov identifikatorja strani podaja naslov, na katerem se nahaja identifikator strani. Tega lahko uporabimo za ugotavljanje ali se na disku nahaja stran podatkovne baze, prav tako lahko s pomočjo le-tega sklepamo o vsebini strani (ali gre za podatek, indeks ali kaj tretjega). Naslov identifikatorja strukture (Structure Identifier Address) podaja identifikator podatkovne strukture, ki mu stran pripada (npr. tabela kupcev). Naslov enoznačnega identifikatorja strani in njegova velikost enoznačno predstavljata podatkovno stran (ta podatek lahko uporabimo za iskanje, kje vse se stran referencira, oz. je v uporabi)

2.1.2 *Parametri imenika vrstice*

Imenik vrstice omogoča vpogled, kje in kako je posamezna vrstica shranjena v strani. Slika 3 prikazuje kako so parametri v njem shranjeni. Imenik vrstice vsebuje seznam naslovov, na katerem se nahajajo posamezne vrstice, dodatno so v imeniku lahko shranjeni tudi seznami spremenjenih vrstic. Naslov vrstice imenika določa lokacijo imenika ali lokacijo prve vrstice v imeniku. Velikost naslova nam določa koliko bajtov je med dvema naslovoma vrstice. Da bi lahko prebrali poljubno vrstico znotraj imenika, potrebujemo položaj prvega naslova (High Value Position) in ustrezni konstanti C_x in C_y.

2.1.3 *Parametri podatkovne vrstice*

Podatki vrstic zasedajo večino prostora v strani, zato je smiselno, da se posvetimo parametrom, ki karakterizirajo

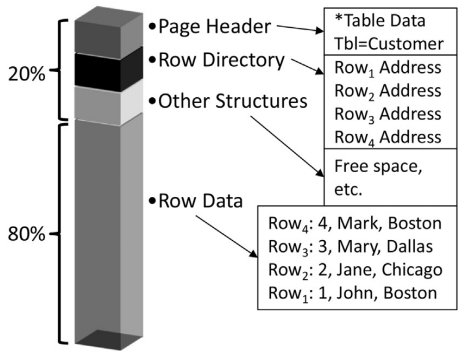


Figure 2: Struktura strani podatkovne baze

podatkovno vrstico. Identifikator vrstice je vsekakor pomemben, ker predstavlja podatek v vrstici, ki ga generira podatkovna baza. Število stolpcev nam pomaga razčleniti vrstico na posamezne vrednosti. Omeniti je potrebno, da število stolpcev ponavadi ni enako številu stolpcev, ki so vidni v tabeli. Podatkovna baza lahko interno doda identifikator vrstice v samo tabelo. Določeni podatkovni tipi potrebujejo dodatne podatke, kot npr. dolžina, velikost, itd. Tovrstni podatki so shranjeni v imeniku stolpca, ki ima podobno vlogo kot imenik vrstice. Slika 4 prikazuje različne kombinacije postavitve imenikov in podatkov vrstice znotraj. Različni produkti uporabljajo različne postavitve, zato je pomembno identificirati vse parametre tako, da niso odvisni od kombinacije postavitve. S slike je tudi razvidno, zakaj so naslovi kot parametri zelo pomembni.

2.2 Interne podatkovne strukture

S stališča forenzike so zanimive ostale strukture, ki jih podatkovna baza uporablja za hitrejši dostop, lažje vzdrževanje podatkov in beleženje sprememb. Število tovrstnih struktur je od implementacije do implementacije različno in presega namen tega članka. Omejimo se le na nekaj najbolj pogostih, začnimo z indeksi.

Indeks je pomožna struktura, ki omogoča hitrejše izvajanje poizvedb nad podatki v podatkovnih tabelah. Predstavljajmo si tabelo kupcev, sortiranih po številki kupca. Če bi iskali kupca po nekem atributu, npr. ime kupca, bi morali preiskati celotno tabelo, da bi našli vse kupce, ki ustrezajo nekem kriteriju. Če bi imeli na voljo indeks - imenik vseh imen kupcev, bi lahko precej učinkoviteje iskali podatkih. Indeksi so stališča forenzike še posebej zanimivi, saj jih podatkovne baze samodejno ustvarijo za vse primarne ključe v tabelah (s tem ohranjajo logično strukturo tabele!). Druga prednost indeksa je njegova drevesna struktura, ki praviloma ne omogoča neposrednih sprememb na listih. Spremembe ni možno pisati neposredno, marveč se doda nova veja, skupaj z listi. Praviloma to pomeni, da se stare vrednosti lahko ohranjajo precej dlje in lahko vsebujejo forenzično zanimive informacije.

Druga tovrstna struktura je materializirani pogled (materialized view). Za razliko od navadnih pogledov, ki se izvedejo na zahtevo in posredujejo podatke, gre v tem primeru za dinamično ustvarjene tabele, ki so fizično shranjeni na disku. Prednost materializiranega pogleda so njegove performanse, saj je lahko iskanje podatkov zamudno, v materializiranem pogledu pa so podatki že na voljo. V teh pogledih se tako nahajajo posnetki podatkov iz tabel, seveda ni nujno, da so podatki povsem ažurni.

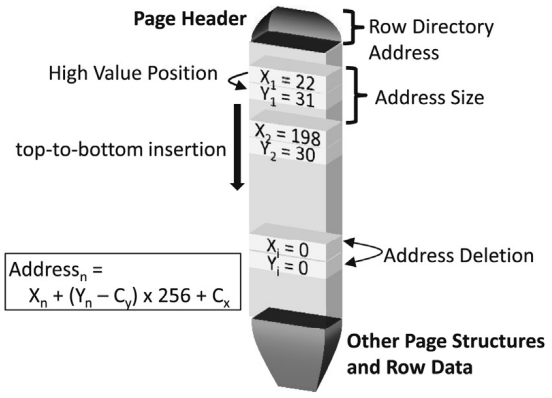


Figure 3: Struktura strani imenika vrstice

Kot zadnji primer lahko navedemo dnevnik transakcij. Transakcije omogočajo sočasni dostop do podatkov in njihovo celovitost pri posodabljanju. S stališča forenzike so lahko zanimive tako uspešne transakcije (ki so spremenile podatke), kot tudi poskusi sprememb ali razveljavitve (rollback) sprememb. Vse tovrstne spremembe pa se beležijo v dnevniku transakcij in se jih lahko uporabi za rekonstruiranje podatkovne baze.

2.3 Začasne podatkovne strukture

Sistemi za upravljanje z bazami podatkov pogosto uporabljajo strukture, ki omogočajo hitrejšo obdelavo podatkov v delovnem pomnilniku. Podatkovne strani, ki se pogosto uporabljajo se prestavijo v delovni pomnilnik, na ta način se znatno skrajša čas obdelave podatkov. Spremembe na vrsticah v strani se tako izvajajo neposredno v pomnilniku, tovrstne strani podatkovna baza označi kot "umazane"(eng. dirty pages). Na neki točki se takšne strani umaknejo s pomnilnika in se zapišejo na disk, pri čemer se stara vsebina prepiše. S forenzičnega stališča so pisanja strani na disk zanimiva, ker se pogosto zapišejo na nezaseden prostor. Prostor, ki ga zasedajo stari podatki se praviloma zgolj označi za nezaseden in ostane na disku.

Začasne spremembe v pomnilniku lahko vplivajo na določene vrste indeksov (v katere se doda neka vrednost), sprememba sama se potem lahko tudi zavrže, vnos v indeksu pa ostane, saj je brisanje vnosov z indeksa precej zamuden proces. Zanimive so tudi spremembe znotraj strani, saj se lahko vrstice, ki so bile izbrisane iz tabele v sami strani zgolj označijo za nedosegljive (in se fizično ne prepišejo).

3. DEKONSTRUKCIJA PODATKOVNE BAZE

V tem poglavju je predstavljeno kako različna je uporaba parametrov med posameznimi implementacijami sistemov za uporabljanje s podatkovnimi bazami in kakšne strategije za shranjevanje podatkov uporabljajo. Orodje, ki so ga razvili avtorji članka, podpira sledeče podatkovne baze: Oracle, PostgreSQL,MySQL, SQLite, Apache Derby, DB2, SQLServer in FireBird

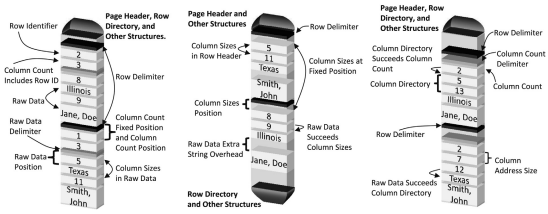


Figure 4: Različni primeri postavitve imenikov in podatkov vrstice znotraj strani

Parameter	Oracle	PostgreSQL	SQLite	Firebird	DB2	SQLServer	MySQL	ApacheDerby
Identifikator strukture	Da	Ne	Da					Ne
Enoznačni ident. strani	Da							Ne
Zap. imenika vrstice	zgoraj-dol					Spodaj-gor		
Identifikator vrstice	Ne	Da	Ne				Da	
Število stolpcev	Da		Ne		Da		Ne	Da
Velikosti stolpcev	Da				Ne		Da	
Imenik stolpca	Ne				Da		Ne	
Števila shranjena z nizi	Da				Ne		Da	

Table 1: Seznam parametrov in kompromisov pri strukturi strani

3.1 Parametri za shranjevanje podatkov - kompromisi

Kot je razvidno s tabele 1, večina implementacij uporablja identifikator strukture, kar olajša iskanje strani na sami sliki diska, ko tudi znatno olajša sestavljanje podatkovne baze iz posameznih strani.

V drugih primerih, kjer identifikator ni na voljo, se je orodje pri rekonstruiranju moralo zanesti na število stolpcev v tabeli. Žal ima ta pristop težavo v primeru, ko se v podatkovni bazi nahajata dve tabeli, ki imata identično število stolpcev posamezni stolpci pa so enakih tipov. V tem primeru se podatki združijo v eno samo tabelo in ni možno ugotoviti, kateri tabeli dejansko vrstica pripada.

Unikatni identifikator strani je prav tako prisoten v veliki večini podatkovnih baz, kar znatno olajša prepoznavanje strani na različnih lokacijah (npr. v pomnilniku in na disku). V nekaterih primerih je identifikator sestavljen iz različnih ključev (npr. identifikator datoteke + identifikator strani), kar je načeloma dobrodošlo, saj nam te informacije lahko pridejo prav.

Zaporedje imenika vrstice je razdeljeno na zgoraj-dol in spodaj-gor. Oboje nam pomaga ugotoviti zaporedje, v katerem so bile vrstice dodane na stran. Identifikator vrstice je v nekaterih implementacijah prisoten, v drugih ga ni, je pa zelo koristen, kadar restavriramo podatke iz zaporedja posodabljanj ali brisanj.

Večina podatkovni baz tudi uporablja številčenje stolpcev, kar olajša razčlenitev vrstic. Brez tega je namreč členitev bistveno bolj zahtevna, saj mora orodje prepoznati shemo vrstice, glej poglavje 3.2. Šele s shemo (oz. identifikatorjem strukture) lahko prepoznamo vse strani. Praktično vse podatkovne baze uporabljanjo enega od zgoraj omenjenih načinov za prepoznavanje strani.

Velikosti stolpcev se tipično uporabljajo v primerih, ko imamo opravka s surovimi podatki. Velikost podatka v stolpcu znatno olajša razčlenitev posameznih vrednosti. Kadar ta informacija ni na voljo neposredno, je dosegljiva v imeniku stolpca. V slednjem primeru tako ni potrebno, so stolpci shranjeni strogo zaporedno, temveč se lahko prepletajo.

3.2 Odkrivanje parametrov

Orodje avtorjev omogoča skoraj samodejno odkrivanje parametrov v podatkovni bazi. Pri testiranju se je uporabljal nabor sintetično generiranih podatkov, kot tudi podatki iz SSBM testa. Orodje mora biti predhodno ustrezno skonfigurirano, pri čemer je morajo biti v konfiguracijski datoteki podane sledeče karakteristike podatkovne baze: velikost strani, datotečni imenik, kjer so shranjene datoteke podatkovne baze, ime podatkovne baze, prijavni podatki (credentials) za podatkovno bazo z dovoljenji za ustvarjanje tabel in zapisovanje podatkov.

V primeru, da gre za podatkovno bazo, ki še ni podprta, je potrebno implementirati razred, ki se poveže na podatkovo bazo in izvede poljubno SQL poizvedbo. Med iskanjem parametrov, se vnosi podatkov izvajajo ročno, ker orodja za masoven uvoz podatkov tipično ne zagotavljajo vrstnega reda vrstic.

Restavriranje sheme podatkovne baze v primeru, da le-ta ni vnaprej znana, terja v nekaterih primerih dodatno delo. Nekatere podatkovne baze namreč ne implementirajo identifikatorja strukture. V takih primerih orodje poskuša uganiti shemo iz dosegljivih podatkov. V kolikor naleti na podatek, ni ne sovpada z obstoječo shemo, orodje poskusi prebrati podatke s posodobljeno shemo. Poleg podatkov, orodje iz podatkovne baze restavrira tudi stolpce, ki vsebujejo unikatne podatke (unikatni oz. primarni ključi).

3.3 Rekonstrukcija začasnih artefaktov

Ko se vsebina v podatkovni bazi spreminja, tako dejanje ustvari kar nekaj sledi, ki ostanejo. Novo nastale podatke lahko tako rekonstruiramo iz zaporedja vnosov in posodobitev, prav tako lahko rekonstruiramo dejanja uporabnika, ki je podatke spreminjal (npr. zaporedje SQL stavkov). Posebej zanimiva je možnost odkrivanja informacij v spremembah, ki so bile preklicane ali zavrnjene (eng. transaction rollbacks). Zadnja kategorija je z vidka forenzike najbolj zanimiva, saj vsebuje informacije, ki uporabnikom podatkovne baze ni na voljo, ne glede na stopnjo pravic, ki jo le-ti imajo.

Praktično še najmanj zanimivi so vnosi informacij (data inserts), saj vsebujejo relativno malo zanimivih informacij, ker se v tem primeru ustvari povsem nova vrstica. Edini zanimivi aspekt v tem primeru je vrstni red vnosa informacij, saj se praviloma nove vrstice zaradi performančnih razlogov dodajajo v obstoječe polprazne strani. Posebej velja omeniti masovne vnose podatkov (bulk inserts), saj jih je možno ugotoviti iz vzorca, kako so bile vrstice zapisane v strani.

Brisanja podatkov so na drugi strani bistveno bolj zanimiva. Tako kot datotečni sistemi označujejo datoteke zgolj za izbrisane, tako tudi podatkovne baze lahko označijo posamezne vrstice kot izbrisane. Strategije in scenariji, kjer podatkovne

baze uporabljajo ta pristop bi presegale ta članek - vsaka implementacija ima svoje posebnosti.

Posodabljanja podatkov so z uporabniškega vidika kombinacija brisanja obstoječe in vnosa nove informacije. V resnici pa podatkovne baze posodabljajo podatke na zelo različne načine. V primeru, ko velikost novega podatka ne presega velikosti obstoječega, se stran označi za zapisovanje, sama vrednost pa se bodisi prepiše z novo ali pa se prepiše kar celotna vrstica. V primeru, ko je nova vrednost večja, od obstoječe se v vseh primerih obstoječa vrstica izbriše in doda nova.

4. EKSPERIMENTALNO DELO

Orodje, ki so ga razvili avtorji članka[1] podpira osem različnih sistemov relacijskih baz (RDBMS) (tabela 2), ki tečejo pod operacijskima sistemoma Linux in Windows. Za potrebe analize delovanja orodja, so najprej pridobili sliki delovnega spomina in trdega diska. Za analizo podatkov iz diska so bile, ali uporabljene datoteke za katere skrbi podatkovna baza, ali neposredno branje iz slike diska.

4.1 Eksperiment 1: Testiranje različnih verzij sistemov podatkovnih baz.

Namen prvega eksperimenta je bil dokazati, da orodje deluje tako na različnih sistemih podatkovnih baz, kot operacijskih sistemih Linux in Windows. Za podatkovne baze v tabeli 2, je bilo pokazano, da je orodje sposobno samodejno prepoznati potrebne parametre z uporabo mehanizma opisanega v prejšnjem poglavju 3.2, in uspešno rekonstruirat podatke iz strani. Prav tako je bilo ugotovljeno, da se podatkovne strukture podatkovnih baz istega proizvajalca med verzijami ne spreminjajo, z izjemo PosgreSQL kjer so se med verzijo 7.3 in 8.4 spremenili parametri kot na primer splošni identifikator strani, naslov imenika vrstice, ... itd. Iz tega sledi, da je mogoče za večino verzij podatkovnih baz uporabiti iste parametre za rekonstrukcijo.

4.2 Eksperiment 2: Rekonstrukcija podatkov vrstic

V tem eksperimentu je bila preizkušena zmožnost orodja rekonstrukcije podatkov iz slik strani podatkovne baze. Postopka za preiskovanje slike diska in delovnega spomina sta ista s to razliko, da se lahko na delovnem spominu stran razlikuje od tistega na disku zaradi posodobitev podatkov.
Ker vsebina medpomnilnika vsebuje informacije katere vrstice so bile pred kratkim dostopane s poizvedbami, so se avtorji odločili to prikazati kar vizualno. Slika 5 prikazuje stanje medpomnilnika podatkovne baze Oracle (50.000 strani), kjer vsaka točka predstavlja eno stran, ter pripadajoči stolpni graf, ki prikazuje število strani na posamezni sliki nad njim.

Na začetku je medpomnilnik napolnjen s sintetičnimi tabelami. Druga slika prikazuje stanje medpomnilnika po izvedbi večjega števila poizvedb s tabelami Customer in Part. Zadnji dve sliki prikazujeta stanje medpomnilnika po zagonu poizvedb s tabelo Lineorer. Tretja slika po 100 poizvedbah in slika 4 po še dodatnih 200 zagonih. Avtorji sicer poudarjajo, da so bile poizvedbe izbrane tako, da so vse strani poizvedb pomnjene v medpomnilnik.

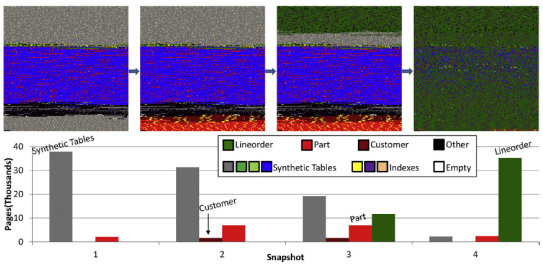


Figure 5: Stanje medpomnilnika po več izvedenih poizvedbah

4.3 Eksperiment 3: Rekonstrukcija pokvarjenih podatkov

Eksperiment se je izvajal na sveže ustvarjeni bazi PostgreSQL v oblaku. Baza je bila napolnjena z SSBM benchmark podatki[2], ki ustvari 24 milijonov vrstic, nato pa je bila podatkovna baza izklopljena ter datoteke, ki vsebujejo podatke podatkovne baze izbrisane.
Ker je izbrisani prostor na disku označen kot prost, ga lahko druge aplikacije prepišejo, kar je bilo simulirano s pisanjem naključnih 1KB velikih zapisov po celotnem disku. Rezultati so prikazani v tabeli 3.
Drug stolpec prikazuje začetno število blokov preden je bila narejena škoda na disku. Tretji in četrti stolpec prikazujeta stanje, ko je bilo narejeno 10% in 25% škode. Točna izguba se na posameznih tabelah razlikuje zaradi naključnosti, vendar je povprečna izguba podatkov dovolj blizu dejanski povzročeni škodi.

5. EKSPERIMENT 4: ODMEVI BRISANJA IZ PODATKOVNE BAZE

V tem eksperimentu so testirali kdaj je podatek dokončno izbrisan iz podatkovne baze. Uporabljena je bila podatkovna baza Oracle v kateri so tabeli kupcev naredili indeks na polju telefonske številke. Tabela 4 prikazuje časovnico, ter podatkovne strukture (tabela, indeks, materializiran pogled) in nosilec kjer se struktura nahaja: HDD - trdi disk; RAM - delovni spomin.
Oznake v tabeli:

- ☎– telefonska številka je še vedno dosegljiva s SQL poizvedbo
- ✓– telefonska številka ni dosegljiva s SQL poizvedbo, vendar jo lahko z orodjem obnovimo
- ✗– telefonska številka ni dosegljiva s SQL poizvedbo, vendar jo lahko z orodjem obnovimo, ter lahko ugotovimo, da je bila označena kot izbrisana

V času:

- T_0 je podatek o telefonski številki prisoten v vseh treh podatkovnih strukturah na trdem disku
- T_1 je bila vrstica s strani uporabnika iz podatkovne baze izbrisana kar povroči, da se stran tabele in stran indeksa shranita v delovni pomnilnik

Verzija PB	OS	Medpomn. (MB)	Velikost strani (MB)
Apache Derby 10.10	Linux	400	4
Apache Derby 10.5	Linux	400	4
DB2 Express-C 10.5	Linux	400	4
Firebird 2.5.1	Linux	400	8
Firebird 2.1.7	Windows	400	8
MySQL Server 5.1.73	Linux	800	16
MySQL Server 5.6.1	Windows	800	16
Oracle 11g R2	Windows	800	8
Oracle 12c R1	Windows	1200	8
PostgreSQL 7.3	Linux	400	8
PostgreSQL 8.4	Linux	400	8
PostgreSQL 9.3	Windows	800	8
SQLite 3.8.6	Linux	2	1
SQLite 3.8.7	Windows	2	1
SQLServer 2008 Enterprise	Windows	800	8

Table 2: Seznam podatkovnih baz uporabljenih v članku

Tabela	Škoda = 0%	Škoda = 10%	Škoda = 25%
Dwdate	35 (100%)	31 (88.6%)	20 (57.1%)
Supplier	565 (100%)	455 (80.5%)	326 (57.7%)
Customer	1915(100%)	1559(81.4%)	1075 (56.1%)
Part	8659 (100%)	6969 (80.5%)	4864 (56.2%)
Lineorder	115K (100%)	104K (89.9%)	87K (75.2%)
Total	416K (100%)	375K (89.9%)	312K (74.9%)

Table 3: Rekonstrukcija pokvarjenih podatkov

- T_2 uporabnik izvede poizvedbo z materializiranim pogledom in povzroči, da se telefonski podatek zapiše v delovni pomnilnik
- T_3 se materializiran pogled osveži. Podatek se izbriše iz delovnega pomnilnika, vendar je mogoče, da se delčki podatkov tam še vedno nahajajo
- T_4 je izvedena serija poizvedb, dovolj, da se medpomnilnik prepiše. Ker je tabela uporabljena s strani uporabnika, ostaja shranjena v delovnem pomnilniku
- T_5 je izvedena serija poizvedb, med katerimi tabela kupcev ni uporabljena. Stran tabele se tako iz delovnega pomnilnika izbriše
- T_6 je izveden ponoven izračun indeksa, kateri povzroči, da se indeks z zbrisano vrstico izbriše iz delovnega spomina
- T_7 je izvedena ponovna izgradnja tabele in tako se zapis z zbrisano vrstico dokončno izbriše iz delovnega spomina

Event	Table		Index		MV	
	HDD	RAM	HDD	RAM	HDD	RAM
T_0	☎		✓		☎	
T_1	✓	✗	✓	✓	☎	
T_2	✓	✗	✓	✓	☎	✓
T_3	✓	✗	✓	✓		
T_4	✗	✗	✓			
T_5	✗		✓			
T_6	✗					
T_7						

Table 4: Časovnica brisanja telefonske številke

6. ZAKLJUčEK IN IZTOčNICE ZA NAPREJ

Orodje, ki so ga razvili avtorji članka predstavlja dobro alternativo plačljivim orodjem, ki so razvita za relacijske podatkovne baze za točno določenega proizvajalca kot so Office Recovery za Oracle[5], PostgreSQL[6] in MySQL[4], Drinkwater za SQLite[3], ter Phoenix Stellar za podatkovne baze IBM DB2[7] in Microsoftov SQL Server[8].
Ker se podatkovne baze istega proizvajalca med verzijami, iz vidika zgradbe posameznih podatkovnih struktur ne razlikujejo, v kolikor poznamo parametre za eno verzijo nam ni potrebno ponovno nastavljati parametrov za drugo. Univerzalno orodje olajša delo tako digitalnim preiskovalcem, kot tudi podjetjem, ki se ukvarjajo z restavriranjem izbrisanih, ali pokvarjenih podatkov podatkovnih baz. Orodja ni bilo mogoče preizkusiti, saj avtorji ne navajajo za katero orodje gre niti, kje bi ga bilo moč najti.

7. REFERENCES

[1] James Wagner, Alexander Rasin, and Jonathan Grier. Database forensic analysis through internal structure carving. *Digital Investigation*, 14(1):S106–S115, 2015.

[2] O'Neil et al. Star Schema Benchmark. `http://www.cs.umb.edu/~poneil/StarSchemaB.PDF`. Accessed: 2016-05-10.

[3] Sausage Factory. Drinkwater R. Forensics from the sausage factory. `http://forensicsfromthesausagefactory.blogspot.`

si/2011/04/carving-sqlite-databases-from.html. Accessed: 2016-05-10.

[4] Office Recovery. OfficeRecovery for MySQL. http://www.officerecovery.com/mysql/. Accessed: 2016-05-10.

[5] Office Recovery. OfficeRecovery for Oracle. http://www.officerecovery.com/oracle/. Accessed: 2016-05-10.

[6] Office Recovery. OfficeRecovery for PostgreSQL. http://www.officerecovery.com/postgresql/. Accessed: 2016-05-10.

[7] Stellar Data Recovery. Stellar Phoenix for DB2 Recovery Software. http://www.stellarinfo.com/database-recovery/db2-recovery.php. Accessed: 2016-05-10.

[8] Stellar Data Recovery. Stellar Phoenix for MS SQL Database repair. http://www.stellarinfo.com/sql-recovery.htm. Accessed: 2016-05-10.

Del V

Forenzika e-pošte in spletnih storitev

Ohranjanje zasebnosti pri forenzični preiskavi sporočil elektronske pošte

Seminarska naloga pri predmetu Računalniška forenzika

Anita Gantar
Univerza v Ljubljani
Fakulteta za matematiko in fiziko
Fakulteta za računalništvo in informatiko
anita.gantar@gmail.com

Anže Nunar
Univerza v Ljubljani
Fakulteta za matematiko in fiziko
Fakulteta za računalništvo in informatiko
anze@nunar.si

Grega Štravs
Univerza v Ljubljani
Fakulteta za matematiko in fiziko
Fakulteta za računalništvo in informatiko
stravs.grega@gmail.com

POVZETEK

Seminarska naloga opiše težave, s katerimi se srečujejo preiskovalci pri analizi sporočil elektronske pošte. Pri tem pogosto nezakonito posežejo v posameznikovo zasebnost. Predstavljen je nov način pregledovanja sporočil elektronske pošte, ki s pomočjo kriptografije ohranja zasebnost posameznika, hkrati pa preiskovalcu omogoča opravljanje forenzične preiskave. V seminarski nalogi je opisan takšen način, njegovo delovanje ter kratek povzetek njegove uporabe.

Ključne besede

analiza, autopsy, digitalna forenzika, elektronska pošta

1. UVOD

Digitalna forenzika nam omogoča uporabo računalniških znanstvenih metod za iskanje odgovorov na pravna vprašanja, ki ponavadi vključujejo pridobitev, analizo in predstavitev digitalnih dokazov. V Republiki Sloveniji je zasebnost posameznika v takšnih preiskavah zaščitena s 37. členom ustave, členom o zagotavljanju tajnosti pisem in drugih občil, ki pravi, da *samo zakon lahko predpiše, da se na podlagi odločbe sodišča za določen čas ne upošteva varstvo tajnosti pisem in drugih občil in nedotakljivost človekove zasebnosti, če je to nujno za uvedbo ali potek kazenskega postopka ali za varnost države* [18]. Podobno določilo zasledimo tudi v splošni deklaraciji človekovih pravic, ki jo je sprejela in razglasila Generalna skupščina Združenih narodov leta 1948. Gre za 12. člen, ki pravi, da *se ne sme nikogar nadlegovati s samovoljnim vmešavanjem v njegovo zasebno življenje, v njegovo družino, v njegovo stanovanje ali njegovo dopisovanje in tudi ne z napadi na njegovo čast in ugled. Vsakdo ima pravico do zakonskega varstva pred takšnim vmešavanjem ali takšnimi napadi* [16].

Eden izmed problemov, ki pogosto nezakonito posežejo v posameznikovo zasebnost, je analiza elektronske pošte, še posebej v primerih, ko je zaposlenim v podjetjem uporaba službenega elektronskega računa prepovedana v zasebne namene. Po mnenju informacijske pooblaščenke zaposleni na delovnem mestu upravičeno pričakujejo določen obseg zasebnosti. Delodajalca, kot upravljavca osebnih podatkov svojih zaposlenih, pri tem omejuje zakonodaja, natančneje 46. in 48. člen, ki pravita, da mora delodajalec spoštovati in varovati delavčevo osebnost ter upoštevati in ščititi njegovo zasebnost ter da se osebni podatki lahko zbirajo, obdelujejo, uporabljajo in posredujejo tretjim osebam samo, če je to določeno z zakonom oziroma če je to potrebno zaradi uresničevanja pravic in obveznosti iz delovnega razmerja ali v zvezi z delovnim razmerjem [19, 20].

Ljudje se v zadnjih letih vse bolj zanimajo za svoje pravice v povezavi z elektronsko pošto in varovanjem zasebnosti. V zadnjem letu so na uradu informacijske pooblaščenke dobili 15 vprašanj v zvezi s pravicami posameznika in varovanjem zasebnosti pri uporabi elektronske pošte, kar je 7,5-krat več kot leto prej [14]. Tudi to je zagotovo eden od pokazateljev, da gre za aktualno področje.

1.1 Motivacija

V primeru, ko podjetje postane žrtev prevare ali je vpleteno v kartelne dogovore, to običajno zahteva obsežne preiskave, v katerih se pregleduje in analizira tudi sporočila vpletenih, ki lahko vsebujejo podatke o komunikaciji med vpletenimi, ki so dragoceni dokazi za preiskavo. Vendar pa poleg tega v njih najdemo tudi druge, zasebne ali celo občutljive zasebne podatke. Zato je zakonska dolžnost, da se zasebnost pregledovanih oseb med preiskavo zaščiti kolikor je to mogoče.

V ta namen se navadno uporabljajo orodja za filtriranje, ki preprečujejo branje vseh sporočil. Preiskovalcem je dovoljeno, da opravijo iskanje na podlagi seznama ključnih besed in pridobijo vsebino samo tistih sporočil, ki vsebujejo vsaj nekaj ključnih besed iz njihovega seznama. Na ta način naj bi pridobili samo sporočila, ki so pomembna za potek preiskave ter čim manj posegli v zasebnost vpletenih.

Ker preiskovalcem ne želimo posredovati celotne baze podatkov, bi morali najprej ponudniku storitev, kjer je elektronska pošta shranjena, poslati seznam ključnih besed. Nato lahko ponudnik pripravi ustrezna sporočila, ki jih vrne preiskovalcem in ti lahko nadaljujejo s preiskavo. Pri tem naletimo na dve težavi:

- kljub temu, da je postopek preiskave tajen, se na tak način s seznamom ključnih besed, ki so pomembne za primer, seznani tudi ponudnik,
- ponudniku ter preiskovalcem povzročimo dodatno delo, potek preiskave pa se podaljša.

V praksi se žal pogosto zgodi, da se zato preiskovalcem posreduje kar celotna podatkovna baza in se jim omogoči neposredno iskanje po njej. Tu pa se znova pojavijo težave, povezane z zasebnostjo: zaradi nevarnosti, da spregledajo kakšno pomembno sporočilo, se preiskovalci pogosto ne poslužujejo filtriranja in pregledajo vsa sporočila.

1.2 Novi pristop

V nadaljevanju bomo podrobneje opisali nov pristop, ki sta ga v letu 2015 predstavila nemška raziskovalca, Frederik Armknecht in Andreas Dewald [3]. Le-ta zagotavlja varovanje zasebnosti vsebine sporočil, čeprav preiskovalcem posredujemo celotno podatkovno bazo in lahko iskanje na njej opravijo sami. Proces iskanja razkrije vsebino elektronskih sporočil samo v primeru, ko se v njej pojavi zadostno število ključnih besed t iz preiskovalčevega seznama. V nasprotnem primeru preiskovalec o vsebini elektronskega sporočila in o morebitni vsebovanosti ključnih besed ne izve ničesar.

Pristop sestavljajo naslednji štirje koraki:

1. korak: ponudnik izvozi vsa sporočila
2. korak: ponudnik šifrira vsa sporočila s kriptosistemom, ki je opisan v nadaljevanju
3. korak: ponudnik izroči podatkovno bazo preiskovalcem
4. korak: preiskovalci sami opravijo iskanje s seznamom ključnih besed in lahko dešifrirajo samo tista sporočila, ki vsebujejo zadostno število ključnih besed t iz seznama

S takšnim pristopom so preiskovalci sposobni izvajati forenzično preiskavo na običajen način, pri čemer se ne ogroža zasebnosti posameznika. Poudariti želimo, da kljub dostopu do celotne baze podatkov in poljubnemu iskanju, ne morejo dešifrirati in prebrati vsebine sporočil, ki niso povezana s preiskavo. Poleg tega pa tudi ponudnik, kjer se nahaja elektronska pošta, ne ve ničesar o ključnih besedah, ki so na preiskovalčevem seznamu.

Dodatno delo, ki ga predstavlja omenjeni pristop, je le določitev pragu t, ki predstavlja število ključnih besed, ki se morajo pojaviti v sporočilu, da menimo, da je sporočilo povezano s primerom in ga mora preiskovalec prebrati. Poleg tega mora preiskovalec pripraviti tudi seznam ključnih besed. Pripravi lahko beli seznam (angl. *whitelist*) ali črni seznam (angl. *blacklist*). Črni seznam vsebuje besede, ki ne omogočajo napredka pri preiskavi. To so na primer zaimki, imena podjetij, pozdravi ali druge splošne besede, kot na primer, predlogi. Beli seznam vsebuje besede, ki so povezane samo s trenutno preiskavo. V nadaljevanju se bomo omejili zgolj na črni seznam, ki ga lahko pripravimo enkrat in uporabimo v več različnih preiskavah. Seveda pa obstajajo primeri, kjer je bolj smiselna uporaba belega seznama.

V nadaljevanju bomo podrobneje predstavili omenjeni novi pristop.

2. KRIPTOSISTEM

Ker moramo preiskovalcu omogočiti dešifriranje in posledično branje sporočil, ki se ujemajo samo z določenimi besedami, se preiskovalec o ključu ne sme naučiti nič. V nasprotnem primeru bi lahko sam dešifriral vsa sporočila in naša zaščita bi bila nepotrebna. Zato bomo šifrirali vsako sporočilo posebej. Čistopis P nam predstavlja vsebino enega elektronskega sporočila, skupaj s seznamom besed

$$w = (w_1, w_2, \ldots, w_3),$$

ki se pojavijo v njem in niso na preiskovalčevem črnem seznamu. Potrebujemo dva algoritma:

- algoritem za šifriranje sporočila $Protect$,
- algoritem za dešifriranje sporočila $Extract$.

Naloga šifrirnega algoritma $Protect$ je spremeniti čistopis P v tajnopis C in onemogočiti preiskovalcu branje čistopisa. Poleg tega ob šifriranju tajnopisu dodajmo še nekaj dodatnih podatkov, ki so nam v pomoč pri dešifriranju. Ti dodatni podatki bodo omogočali dešifriranje le v primeru, ko sporočilo vsebuje vsaj t ključnih besed iz preiskovalčevega seznama.

2.1 Algoritem za šifriranje sporočila

V nadaljevanju bomo predstavili algoritem za šifriranje sporočila $Protect$ (algoritem 1).

Algoritem 1 Algoritem za šifriranje sporočila $Protect$

Vhod: Čistopis P, množica n besed $\mathcal{W}$, prag t
Izhod: Tajnopis C, preslikava F

Korak P.1: Šifriranje sporočila

1: Izberemo naključni ključ $K \in \{0,1\}^{\ell_K}$
2: Šifriramo čistopis P v tajnopis C s skrivnim ključem K:

$$C \coloneqq Enc_K(P)$$

Korak P.2: Razbitje skrivnega ključa na deleže

3: Ključ K s pomočjo algoritma $Split$ pri pragu t razdelimo na n deležev:

$$(s_1, \ldots, s_n) \coloneqq Split(K, n, t)$$

kjer je $s_1, \ldots, s_n \in \{0,1\}^{\ell_K}$.

Korak P.3: Povezava ključnih besed in deležev ključa

4: Ustvarimo bijektivno preslikavo F:

$$F(w_i) = s_i$$

za vse $i = 1, \ldots, n$.

5: **return** tajnopis C in preslikavo (C, F)

Sestavljen je iz treh delov:

1. korak predstavlja klasično simetrično šifriranje sporočila. To pomeni, da z naključnim ključem K zašifriramo čistopis P. To nam omogoča, da lahko tajnopis C dešifriramo, če poznamo šifrirni ključ K.

2. korak razdeli tajnopis C na več deležev. S tem bomo zagotovili dešifriranje le ob zadostnem pojavu ključnih besed. V tem koraku uporabimo shemo za deljenje skupne skrivnosti, ki nam omogoči, da rekonstruiramo šifrirni ključ K za pare (t, s_i). Šifrirni ključ K je sestavljen iz n deležev, pri čemer posamezni delež predstavimo kot s_i, kjer vsak delež predstavlja posamezno besedo sporočila. Operacija razcepa nam omogoča, da ob pojavu t ključnih besed lahko rekonstruiramo celoten šifrirni ključ.

3. korak razdeli sporočilo na posamezne besede. To je glavna značilnost novega, predlaganega algoritma. V tem koraku uporabimo funkcijo F, ki slika posamezno besedo w_i v s_i. Torej $F(w_i) = s_i$ za vse $i = 1, \ldots, n$. To uporabniku, ki pozna t različnih ključnih besed $w_{i_1}, \ldots, w_{i_t} \in W$ zagotavlja, da z uporabo funkcije F izpelje ustrezen

$$s_{i_1} = F(w_{i_1}), \ldots, s_{i_t} = F(w_{i_t}).$$

Skrivni ključ K obnovimo kot

$$K = Recover(S_{i_1}, \ldots, S_{i_t}),$$

kar nam omogoča dešifriranje $P = Dec_K(C)$.

Izhod algoritma za šifriranje sporočila *Protect* je šifrirano besedilo, tajnopis $C = Enc_K(P)$ in informacija, ki enolično določa funkcijo F. Zaradi učinkovitosti je potrebno, da ima funkcija F kompaktno predstavitev. Postopka, omenjena v 3. koraku, sta podrobneje opisana v nadaljevanju.

2.2 Algoritem za dešifriranje sporočila

Glavna naloga algoritma za dešifriranje sporočila (algoritem 2) je omogočiti preiskovalcu dešifriranje tajnopisa. To lahko povemo tudi drugače: če v šifriranem sporočilu najdemo več kot t ključnih besed $w'_1, \ldots, w'_t$ lahko s pomočjo preslikave F tajnopis C dešifriramo in dobimo čistopis P.

Tudi algoritem za dešifriranje sporočila sestavljajo trije koraki, ki v obratnem vrstnem redu opravljajo podobne naloge kot v algoritmu za šifriranje sporočila (algoritem 1). Podrobneje si oglejmo posamezne korake algoritma za dešifriranje sporočila:

1. korak predstavlja izračun možnih deležev, kjer z uporabo preslikave F iz ključnih besed $w'_1, \ldots, w'_t$ izračunamo t možnih deležev:

$$s'_i \coloneqq F\left(w'_i\right) \text{ za } i = 1, \ldots, t.$$

2. korak iz izračunanih deležev $s'_1, \ldots, s'_t$ obnovi šifrirni ključ K. Na deležih $s'_1, \ldots, s'_t$ uporabimo algoritem *Recover* iz sheme za deljenje skrivnosti in dobimo vrednost K'. Vidimo lahko, da bo preslikava F vrnila ustreznih t deležev skritega ključa le v primeru, ko se je v šifriranemu sporočilu res pojavilo t ključnih besed. V nasprotnem primeru, ko se pojavi manj kot t ključnih besed ali pa ujemanj sploh ni, nam algoritem vrne neko naključno vrednost K', pri kateri ne izvemo katere iskane ključne besede so bile sicer v sporočilu najdene. Izračunani šifrirni ključ K' je enak šifrirnemu ključu K natanko tedaj, ko velja $\{s'_1, \ldots, s'_t\} \subset \{s_1, \ldots, s_n\}$.

3. korak iz tajnopisa C dešifrira prvotni čistopis P. Izračunamo torej vrednost kandidata: $P' = Dec_{K'}(C)$. Na podlagi splošnih zahtev pri varnih šifrirnih algoritmih vemo, da velja: $P' = P$, če velja $K' = K$. V kolikor slednje ne bi veljalo, bi P' lahko vseboval naključne nesmiselne podatke, ki očitno niso povezani s čistopisom P. Uporabimo učinkovit algoritem, ki zavrne napačne čistopise in v takih primerih vrne $\perp$. V nasprotnem primeru je zadnji izhod algoritma iskani čistopis P. Algoritem v zadnjem koraku preveri ali je K' ustrezen, da preiskovalcu ne izpiše nesmiselnih podatkov.

Algoritem 2 Algoritem za dešifriranje sporočila *Extend*

Vhod: Tajnopis C, preslikava F, t ključnih besed $w'_1, \ldots, w'_t$
Izhod: Čistopis P'

Korak E.1: Izračun možnih deležev
for all $i = 1, \ldots, t$ **do**
    S preslikavo F na i-ti ključni besedi w'_i dobimo delež:

$$s'_i \coloneqq F\left(w'_i\right)$$

end for

Korak E.2: Določitev možnih šifrirnih ključev
t kandidatov za deleže, ki pripadajo K, uporabimo v obnovitvenemu algoritmu iz sheme deljenih skrivnosti, s katerim poiščemo kandidate za skrite ključe:

$$K' \coloneqq Recover\left(s'_1, \ldots, s'_t\right)$$

Korak E.3: Dešifriranje šifriranega čistopisa
Na tajnopisu C uporabimo dešifrirni algoritem z možnim šifrirnim ključem, da dobimo možen čistopis P':

$$P' = Dec_{K'}(C)$$

if P' je veljaven čistopis **then**
    return čistopis P'
else
    return $\perp$
end if

2.3 Podroben opis komponent

V tem podpoglavju bomo podrobneje predstavili uporabljene kriptografske temelje. Naš kriptosistem za svoje delovanje potrebuje naslednje komponente:

- shemo za šifriranje/dešifriranje, ki omogoča učinkovito prepoznavanje napačnih dešifriranih sporočil,
- shemo za deljenje skrivnosti,
- preslikavo F.

V nadaljevanju so podrobneje opisane posamezne komponente, skupaj z izbiro konkretnih algoritmov in parametrov.

2.3.1 Shema za šifriranje/dešifriranje

Pri izbiranju šifrirne sheme se omejimo na take, ki so bile dobro preverjene, uporabljene v daljšem časovnem obdobju in so dovolj učinkovite. Dober kandidat je AES [9], ki je trenutno standard šifriranja.

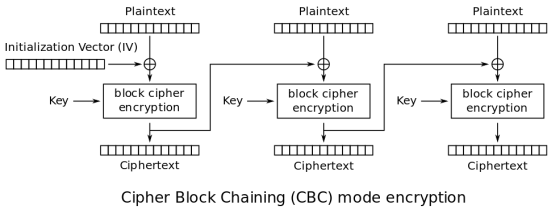
AES je bil ustvarjen za šifriranje blokov velikosti 128 bitov. V kolikor želimo šifrirati večje količine podatkov AES uporabimo v kombinaciji s kakšno drugo obliko šifriranja, kot je na primer veriženje kodnih blokov CBC (angl. *Cipher Block Chaining*). Čistopis P razdelimo v več blokov velikosti 128 bitov $P = (p_1, \ldots, p_\ell)$, kjer je $p_i \in \{0,1\}^{128}$ za vsak i. V kolikor velikost čistopisa P ni večkratnik števila 128, ga na koncu bitno dopolnimo (angl. *padding*). Glede na velikost šifrirnega ključa AES standard podpira 128, 192 in 256 bitno šifriranje. Najboljši znani napad na 128 bitno AES šifriranje ima časovno zahtevnost približno $2^{126,1}$ korakov [26], kar je preveč za praktično izvedbo. Ker je še vedno dovolj varno, priporočamo uporabo slednjega. V prihodnje bo dolžina ključa $\ell_K = 128$ in Enc se bo nanašal na AES-128 in CBC, seveda so možne tudi druge oblike šifriranja.

Oglejmo si šifriranje (slika 1) in dešifriranje (slika 2). Čistopis P najprej razdelimo v zaporedje blokov dolžine 128 bitov $(p_1, \ldots, p_\ell)$. Začetek zaporedja dopolnimo s 128-bitnim blokom sestavljenim iz poljubnih prepoznavnih bitov, npr. niz s samimi ničlami $= \langle 0, \ldots, 0 \rangle$. S tem blokom bomo kasneje lažje prepoznali napačno šifrirane čistopise. Nato s 128-bitnim ključem $K \in \{0,1\}^{128}$ šifriramo zaporedje 128-bitnih blokov $(p_0, p_1, \ldots, p_\ell)$ v tajnopis $(c_0, \ldots, c_\ell)$. V CBC uporabimo naključno izbran inicializacijski vektor IV, potem pa izračunamo šifrirne bloke na naslednji način:

$$c_0 \coloneqq Enc_K\left(pad \oplus IV\right),$$

$$c_i \coloneqq Enc_K\left(p_i \oplus c_{i-1}\right),\ i = 2, \ldots, \ell.$$

Izhod šifriranja je $(IV, c_0, \ldots, c_\ell)$.



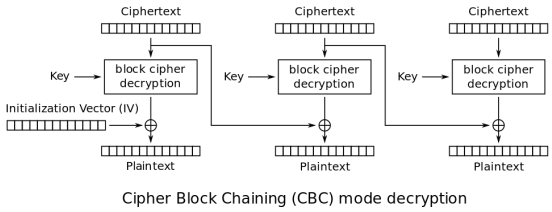
Slika 1: Primer CBC šifriranja [27].

Glede na opisani postopek za dešifriranje velja naslednje:

$$pad \coloneqq Dec_K\left(c_0\right) \oplus IV,$$

$$p_i \coloneqq Dec_K\left(c_i\right) \oplus c_{i-1},\ i = 2, \ldots, \ell.$$

Vidimo lahko, da bo preiskovalec zelo verjetno moral poskusiti več različnih izbir ključnih besed, preden bo iz tajnopisa lahko pridobil izvorni čistopis. Ravno zato smo pri šifriranju na začetek zaporedja dodali blok iz poljubnih prepoznavnih



Slika 2: Primer CBC dešifriranja [27].

bitov, da se dešifriranje ustavi takoj, ko je mogoče oziroma ob uporabi napačnega ključa. S podanimi šifriranimi bloki $(c_0, \ldots, c_\ell)$ in kandidatom za ključ K' lahko preiskovalec iz naslednje enačbe

$$pad \coloneqq Dec_K\left(c_0\right) \oplus IV$$

ugotovi, ali velja $K = K'$. V kolikor je slednje izpolnjeno, dešifriranje nadaljujemo. Sicer postopek na trenutnemu sporočilu končamo in preverimo ali naslednje sporočilo iz množice vsebuje t iskanih terminov.

2.3.2 Shema za deljenje skrivnosti

Postopek deljenja skrivnosti v kriptografiji poznamo že zelo dolgo. V našem predlogu uporabimo shemo za deljenje skrivnosti (angl. *secret sharing scheme*, krajše SSS), ki jo je ustvaril Shamir [17] in uporablja Lagrangeovo polinomsko interpolacijo. Oblika sheme vpliva na konstrukcijo preslikave, zato bomo predstavili njene osnovne ideje.

Naj bo $K \in \{0,1\}^{128}$ skrivni ključ in naj bo element končnega obsega $GF(2^{128})$. Naj bo t prag in naj n predstavlja število deležev (enako številu različnih besed v sporočilu, ki niso na črni listi). Želimo poiskati n deležev $s_1, \ldots, s_n$ takih, da bo poljubna izbira t deležev, kjer je $t \leq n$, omogočila rekonstrukcijo celotnega skrivnega ključa K. Naj bo danih $n+1$ paroma različnih vrednosti $x_0, \ldots, x_n \in GF(2^{128})$, ki predstavljajo rezultat izračunov iz razdelka Ustvarjanje preslikave. Naj bo x_0 naključno izbran iz $GF(2^{128}) \backslash \{x_1, \ldots, x_n\}$. Te vrednosti bodo kasneje predstavljale x-koordinate.

Izberemo polinom $p\left(x\right)$ stopnje $t-1$, za katerega velja

$$p\left(x_0\right) = K.$$

n deležev izračunamo na naslednji način:

$$s_i \coloneqq (x_i, y_i) \in GF(2^{128}) \times GF(2^{128}),$$

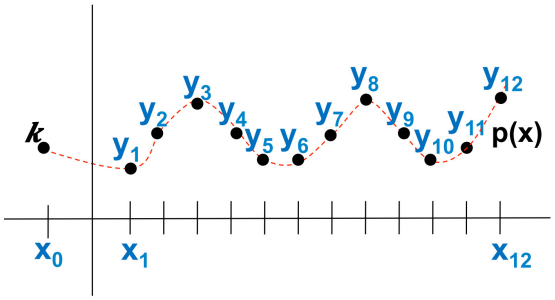
kjer je $y_i = p\left(x_i\right)$. Zapišimo celoten postopek:

$$(s_1, \ldots, s_n) \coloneqq Split\left(K, x_1, \ldots, x_n, t\right).$$

Vidimo lahko, da vsak delež prestavlja eno oceno polinoma p v različni točki. Vemo, da lahko polinom stopnje $t-1$ določimo iz poljubnih t točk. Iz tega dejstva lahko zapišemo še postopek obnovitve: imamo t različnih deležev $s_{i_j} = \left(x_{i_j}, p\left(x_{i_j}\right)\right)$. Najprej interpoliramo polinom $p(x)$ (slika 3). Nato lahko določimo skrivnost na podlagi izračunane vrednosti polinoma $p(x)$ v točki x_0: $K \coloneqq p(x_0)$. Če poznamo manj deležev ali pa je med njimi vsaj en napačen (to pomeni, da ima obliko (x, y'), kjer je $y' \neq p(x)$), potem je izračunana vrednost neodvisna od K. Torej res izpolnjujemo pogoje, ki smo jih omenili v algoritmu. Postopek

zapišemo na naslednji način:

$$K \coloneqq Recover\left(s_{i_1}, \ldots, s_{i_t}\right).$$



Slika 3: Interpolacija polinoma [3].

2.3.3 *Ustvarjanje preslikave*

Sedaj bomo opisali konstrukcijo preslikave F, ki preslika ključne besede $w_i \in \{0,1\}^*$ v skrite deleže $s_i \in \{0,1\}^{256}$. Predlagamo preslikavo F, ki je predstavljena v naslednjih dveh korakih:

1. Uporabimo kriptografsko varno zgoščevalno funkcijo $H : \{0,1\}^* \to \{0,1\}^{256}$. Za H lahko uporabimo na primer SHA-256 [12] ali pa eno izmed oblik SHA-3 [4], ki vrne 256 bitov. Izberemo tudi naključno vrednost $sol \in \{0,1\}^*$ (angl. *salt*), ki nam omogoča dodatno zaščito pred napadi. Nato izračunamo povzetke h_i iz združenih (angl. *concatenated*) bitnih nizov $sol||w_i$, to je:

   $$h_i = H\left(sol||w_i\right) \;\; \forall i = 1, \cdots, n.$$

   Vidimo, da je $h_i \in \{0,1\}^{256}$. Dobljene vrednosti razčlenimo: $(x_i, z_i) \in GF(2^{128}) \times GF(2^{128})$. Vrednosti $\{x_1, \ldots, x_n\}$ bodo x-koordinate, kot smo omenili v razdelku Shema za deljenje skrivnosti. Zahtevamo, da so paroma različne. Trk je par (x_i, x_j); $i \neq j$ z enakim povzetkom. Ker ima vsak x_i dolžino 128 bitov, velja za poljuben par (x_i, x_j), da je verjetnost za trk približno 2^{-128}, če imamo varno zgoščevalno funkcijo. Iz paradoksa rojstnega dneva lahko ocenimo skupno verjetnost kot $n^2/2^{128}$. To verjetnost lahko v praksi zanemarimo, saj je n navadno po velikosti veliko manjši kot 2^{64}. V kolikor pa se zgodi, da sta vrednosti enaki, se korak ponovi z drugo vrednostjo sol.

2. Predpostavimo, da se je shema za deljenje skrivnosti izvedla na podlagi vrednosti $x_1, \ldots, x_n$, ki so določene iz rezultatov $h_i = H\left(salt||w_i\right) \;\; \forall i = 1, \cdots, n$. Imamo n povzetkov $h_i = (x_i, z_i) \in GF(2^{128}) \times GF(2^{128})$ in n deležev $s_i = (x_i, y_i) \in GF(2^{128}) \times GF(2^{128})$. Končajmo konstrukcijo preslikave F. Najprej konstruiramo funkcijo g, ki izpolnjuje naslednji pogoj:

   $$g\left(x_i\right) = y_i \oplus z_i \;\; \forall i = 1, \ldots, n.$$

   Če podamo tako funkcijo g, lahko ustvarimo bijektivno preslikavo

   $$G : GF(2^{128}) \times GF(2^{128}) \to GF(2^{128}) \times GF(2^{128})$$

   $$(x, y) \mapsto (x, g(x) \oplus y)$$

   Za preslikavo G lahko enostavno izračunamo inverz in posledično je bijektivna. Še več, funkcija preslika povzetek $h_i = (x_i, z_i)$ v

   $$G\left(h_i\right) = G\left(x_i, z_i\right) = \left(x_i, g\left(x_i\right) \oplus z_i\right),$$

   $$G\left(h_i\right) = \left(x_i, \left(y_i \oplus z_i\right) \oplus z_i\right),$$

   $$G\left(h_i\right) = \left(x_i, y_i\right) = s_i.$$

   Vidimo, da je F definiran kot kompozitum preslikav H in G, kar zapišemo kot $F = G \circ H$.

Algoritem 3 Preslikava F

Vhod: Ključna beseda $w \in \{0,1\}^*$, zgoščevalna funkcija $H : \{0,1\}^* \to \{0,1\}^{256}$, poljuben salt $salt \in \{0,1\}^*$ in polinom $g(x)$
Izhod: Možen delež $s = (x, y)$

1: Izračunamo $h \coloneqq H\left(salt||w\right)$ pri $h \in \{0,1\}^{256}$
2: Razčlenimo $h = (x, z) \in GF\left(2^{128}\right) \times GF\left(2^{128}\right)$
3: Izračunamo $s \coloneqq (x, z \oplus g(x))$
4: **return** Možen delež s

Algoritem 3 povzema vse korake za konstrukcijo preslikave F. Ker smo predlagali uporabo znane in dobro preverjene zgoščevalne funkcije H lahko opustimo njeno specifikacijo v izhodu algoritma Protect. Edina informacija, ki je potrebna za izračun preslikave F je funkcija $g(x)$. Uresničitev, ki jo predlagamo kasneje, potrebuje zgolj n elementov obsega, kar je v splošnem optimalno. Bijektivnost funkcije F zagotavlja, da se samo pravilni povzetki preslikajo v veljavne deleže.

Poglejmo si še, kako lahko učinkovito uresničimo preslikavo $g : GF(2^{128}) \to GF(2^{128})$. Vemo, da je lahko poljubna preslikava iz končnega obsega v samega sebe predstavljena s polinomom. Torej je najbolj očitno, da uporabimo terke $(x_i, y_i \oplus z_i)$ za interpolacijo polinoma $p(x)$ stopnje w, za katerega velja $p(x_i) = y_i \oplus z_i$ za $i = 1, \ldots, n$. Asimptotična zahtevnost intepolacije ima kubično stopnjo (v našem primeru n). Iz implementacije je vidno, da je tak pristop prepočasen tudi v povprečnem primeru.

Uporabili bomo drugačen pristop. Vhodni prostor (v našem primeru $GF(2^{128})$) bomo razdelili na več različnih množic in za vsakega definirali posamezno 'pod-funkcijo'. Za dani vhodni x izračunamo funkcijo g v x tako, da določimo podmnožico množice $GF(2^{128})$, ki vsebuje x, in nato uporabimo na x-u ustrezno pod-funkcijo. Ustrezne pod-funkcije izračunamo z navadno interpolacijo. Sedaj vsaka množica vsebuje veliko manj vrednosti x_i kot w, s tem pa se zahtevnost zelo zmanjša. To lahko dokažemo. Označimo z ℓ število množic in zaradi lažje predstave predpostavimo, da vsaka množica vsebuje n/ℓ vrednosti x_i. Sledi, da je skupna zahtevnost za interpolacijo teh ℓ pod-funkcij stopnje n/ℓ enaka $\ell \cdot (n/\ell)^3 = n^3/\ell^2$. Če primerjamo to s prejšnjo zahtevnostjo, vidimo da je nov pristop hitrejši za približno faktor ℓ^2.

Za razdelitev $GF(2^{128})$, to je množice vseh 128-bitnih nizov, predlagamo uporabo k najmanj pomembnih bitov (angl. *least significant bits*), kjer je $k \geq 1$ poljubno pozitivno celo število. Natančneje, za vrednost $x \in GF(2^{128})$, z $x\,[1 \ldots k]$

označimo prvih k najmanj pomembnih bitov. S tem dobimo naravno razdelitev $GF(2^{128})$ v 2^k paroma različnih podmnožic. Dva elementa $x, x' \in GF(2^{128})$ pripadata isti množici natanko tedaj, ko imata enakih k najmanj pomembnih bitov, to je $x[1\dots k] = x'[1\dots k]$. Naj $X \coloneqq \{x_1, \dots, x_n\}$ označuje n vrednosti, ki jih določa $h_i = H(sol||w_i)\ \forall i = 1, \cdots, n$. Za $I \in \{0,1\}^k$ definiramo

$$X_I \coloneqq \{x \in X | x[1\dots k] = I\}.$$

Dobimo razdelitev množice X v 2^k paroma različnih podmnožic: $X = \bigcup_{I\in\{0,1\}^k} X_I$. Za vsako izmed množic X_I določimo ustrezno pod-funkcijo g_I, ki izpolnjuje pogoj:

$$g_I(x_i) = y_i \oplus z_i\ \forall x_i \in X_I.$$

Kot smo že omenili, to naredimo z osnovno interpolacijo. Če $n_I \coloneqq |X_I|$ označuje velikost X_I, lahko funkcijo g_I uresničimo s polinomom stopnje n_I, ki bo v povprečju blizu $n/2^k$.

To zaključuje konstrukcijo funkcije g in posledično tudi G. Opredelimo še pod-funkcije g_I, te so:

$$g \hat{=} \left(g_{(0\dots00)}, g_{(0\dots01)}, \dots, g_{(1\dots11)}\right).$$

kjer vsak indeks predstavlja k-bitni niz. Vsak g_I ima stopnjo n_I in ga lahko opišemo z n_I koeficienti v $GF(2^{128})$. Iz $\sum_{I\in\{0,1\}^k} n_I = n$ sledi, da lahko g določimo z n elementi obsega.

Izračun funkcije g je tak, kot je razložen zgoraj. Iz podanega vhodnega $x \in GF(2^{128})$ določimo prvih $I \coloneqq x[1\dots n]$. Nato izračunamo $g_i(x)$, ki predstavlja končni izhod, to je $g(x) \coloneqq g_l(x)$, kjer je $I \coloneqq x[1\dots n]$.

3. IMPLEMENTACIJA

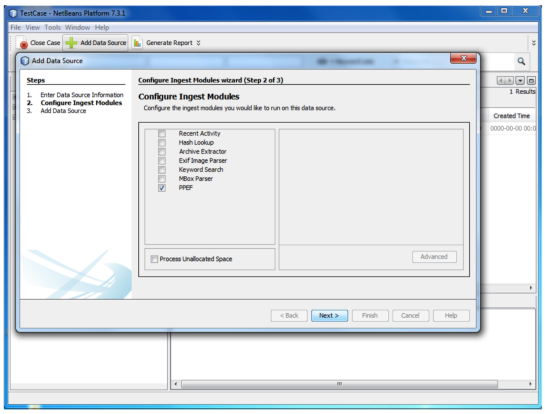
V nadaljevanju je predstavljena implementacija prototipa šifrirnega orodja in integracija iskanja ter dešifriranja e-poštnih sporočil v forenzičnem orodju Autopsy [6]. Na vzorčnih primerih si bomo pogledali kakšna je zmogljivost orodja.

3.1 Realizacija

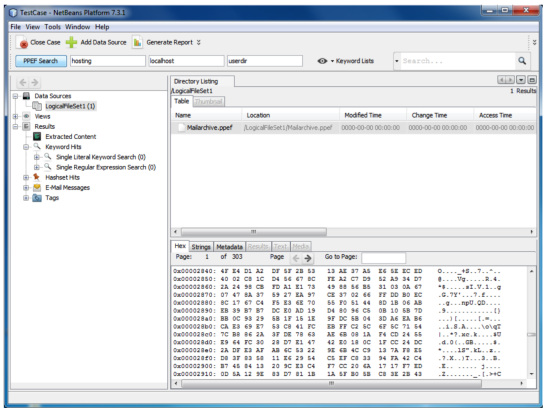
Najprej si oglejmo program za šifriranje e-poštnih sporočil, ki je implementiran v Python skripti in je neodvisen od platforme. Sintaksa ukaza je sledeča:

```
python ppef_gen.py <pst|mh|mbox|rmbox> <in>
       <out> <thr> <blist>
```

Orodje podpira različne formate e-poštnih predalov, kot je naprimer mbox format e-poštnega klienta Mozilla Thunderbird, pst format klienta Microsoft Outlook, MH format ter Maildir format, ki ga uporabljajo različna UNIX orodja. Kot je bilo že povedano, moramo orodju podati vhodne parametre in sicer prag, ki določa število ujemanj ključnih besed v sporočilu, ter črno listo besed, ki je lahko napisana kar v običajni tekstovni datoteki. Program za dešifriranje je prav tako implementiran v Python skripti. Za lažjo uporabo je rešitev intergrirana v orodje Autopsy kot vtičnik imenovan PPEF - Privacy Preserving Email Forensics. Uporabnik v Autopsy preprosto uvozi šifrirano ppef datoteko (Slika 4), jo doda k primeru in izbere PPEF modul, kar omogoča nadaljnje delo s šifriranimi sporočili.



Slika 4: Uvoz šifrirane ppef datototeke v Autopsy [3].



Slika 5: Iskanje s ključnimi besedami na šifrirani ppef datoteki v Autopsy [3].

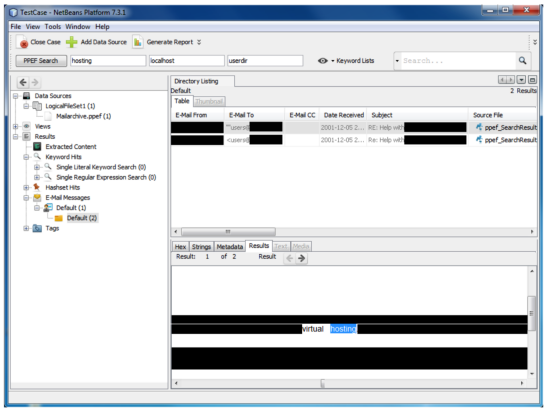
Ko je ppef datoteka uvožena, lahko uporabnik vnese ključne besede in išče po šifrirani vsebini e-pošte (Slika 5). Tako se v ozadju zažene algoritem za iskanje in uspešno dešifrirana sporočila se avtomatsko uvozijo v Autopsy, kjer so potem razčlenjena, da jih lažje pregledujemo (Slika 6).

3.2 Evaluacija

S pomočjo različnih e-poštnih predalov iz realnega sveta, bo v nadaljevanju opisana ocenjena učinkovitost prototipa pri šifriranju, iskanju ter dešifriranju. Za testiranje je bil uporabljen prenosni računalnik z Intel® Core™ i5-3320M procesorjem pri frekvenci 2.60 GHz in 16GB delovnega spomina. Pri testih se je uporabljalo samo eno jedro procesorja.

3.2.1 Testni podatki

1. Apache: Apache httpd e-poštni seznam od 2001 do danes, kjer je 75724 e-poštnih sporočil
2. Work: e-poštni predal z 1590 službenimi e-poštnimi sporočili
3. A, B, C: trije privatni e-poštni predali z 511, 349 in 83 e-poštnimi sporočili



Slika 6: Pregledovanje uspešno dešifriranih sporočil [3].

3.2.2 *Črni seznam besed*

Črni seznam besed je sestavljen iz 10000 najbolj pogostih besed angleškega, nemškega, nizozemskega in francoskega jezika [11], prag ključnih besed pa je nastavljen na 3. Pri realnih primerih mora biti črni seznam besed zaradi večje zaščite pripravljen bolj natančno. To pomeni, da mora vsebovati pogosto uporabljene izraze, naprimer ime podjetja, ki se lahko pojavi v vsakem sporočilu oziroma podpisu. Pri tem moramo upoštevati tudi to, da ne vključimo besed, ki so relevantne našemu primeru. Zato lahko izdelavo črnega seznama ključnih besed gledamo kot ločen problem, ki je zelo odvisen od posameznega primera in ga ni mogoče sestaviti na splošno.

3.2.3 *Leksikografske podrobnosti testnih podatkov*

Tabela 1 opisuje leksikografske podrobnosti testnih podatkov, kjer vsako sporočilo vsebuje okrog 220 besed od tega povprečno 35 ključnih besed.

	Apache	Work	A	B	C
E-pošta	75724	1590	511	349	83
Besede					
Max	77073.00	6186.00	3450.00	1771.00	793.00
Avg	251.34	205.34	196.40	164.00	280.04
Med	174.00	131.00	134.00	62.00	238.00
σ	582.24	325.11	231.48	273.75	125.47
Σ	19032845.00	326486.00	100358.00	57444.00	23243.00
Ključne besede					
Max	3004.00	443.00	536.00	359.00	135.00
Avg	18.74	37.65	33.65	31.00	52.06
Med	15.00	28.00	24.00	15.00	48.00
σ	22.50	38.82	39.24	50.52	20.33
Σ	78914.00	18184.00	2083.00	5265.00	1565.00

Tabela 1: Leksikografske podrobnosti testnih e-poštnih predalov [3].

3.2.4 *Zmogljivost šifiranja*

V tabeli 2 je prikazan čas, ki je potreben za šifriranje testnih sporočil in posameznih e-poštnih predalov. Šifriranja e-poštnega predala ima hitrost približno 13.5 sporočil na sekundo, kar je še sprejemljivo, ker šifriranje zaženemo le enkrat.

3.2.5 *Presežek prostora pri šifriranju*

	Apache [s]	Work	A	B	C
Min	0.004	0.005	0.005	0.005	0.005
Max	31.745	1.403	1.932	1.117	0.460
Avg	0.082	0.136	0.122	0.110	0.173
Med	0.072	0.115	0.101	0.074	0.150
σ	0.133	0.120	0.132	0.153	0.071
Σ	6243.511	217.242	62.842	38.535	14.367

Tabela 2: Zmogljivost enkripcije [3].

Pri samem šifriranju največ zavzetega prostora predstavljata prostor za koeficiente in polinome, ki preslikajo zgoščene vrednosti v deleže. Njihova količina je povezana s ključnimi besedami in polnilom koeficientov pod-funkcij. To povprečno znese 582.4 bajte za polinomske koeficiente in 16 bajtov soli za vsako sporočilo. AES enkripcija pridoda od 33 do 48 bajtov, od tega 16 bajtov za IV in polnilo besedila, ter do 16 bajtov PCKS#7 polnila za AES 128 velikost bloka. Če primerjamo velikost originalnega sporočila in ppef formata pridemo do zaključka, da je velikost ppef formata večja za približno 5.2%.

	Apache	Work	A	B	C
Velikost [KB]	376,551	418,680	16,386	47,486	6,676
Velikost PPEF [KB]	418,870	420,418	16,885	47,821	6,806
Presežek	11.2%	0.4%	3.0%	0.7%	1.9%

Tabela 3: Primerjava velikosti [3].

3.2.6 *Iskanje in dešifriranje*

V tabeli 4 je prikazan čas za dešifriranje pri dovolj velikem številu ujemanjna ključnih besed. Čeprav je povprečen čas za iskanje po posameznem e-poštnem predalu 98 sporočil na sekundo, so iskanja povsem izvedljiva. Prototip je bil ocenjen tudi pri funkcionalni pravilnosti. Izkazalo se je, da je iskanje in ujemanje ključnih besed popolno, kot se je pričakovalo. To pomeni, da so bila dešifrirana le tista sporočila, ki so vsebovala dovolj iskanih ključnih besed.

	Apache [s]	Work	A	B	C
Min	0.0090	0.0096	0.0098	0.0097	0.0098
Max	0.0598	0.1645	0.0139	0.1508	0.0148
Avg	0.0115	0.0137	0.0114	0.0123	0.0117
Med	0.0115	0.0117	0.0113	0.0113	0.0116
σ	0.0007	0.0103	0.0007	0.0086	0.0009
Σ	876.8591	21.7977	5.8650	4.2982	0.9750

Tabela 4: Zmogljivost iskanja in dešifriranja [3].

3.2.7 *Odpornost proti napadom*

V nadaljevanju se bomo posvetili napadom, kjer napadalec s pomočjo slovarja izvede napad na celoten šifriran e-poštni predal, tako da je v povprečju delež π vseh sporočil dešifriran, kjer je $0 \leq \pi < 1$. Najprej si poglejmo napad z grobo silo, kjer z q označimo število poskusov, ki jih mora napadalec narediti. Pogoj za dešifriranje enega e-poštnega sporočila z verjetnostjo $\geq \pi$:

$$q \geq \log_2(1-\pi)/\log_2\left(1-\frac{\binom{n}{t}}{\binom{N}{t}}\right).$$

V tabeli 5 so prikazane zahtevnosti napadov pri različnih pogojih, kjer vsako sporočilo vsebuje n_{avg} ključnih besed za

vsak e-poštni predal. Za število vse možnih uporabljenih besed za primer vzamemo angleški slovar Oxford English Dictionary, ki vsebuje 171,476 besed, ki so trenutno v uporabi [15]. Za začetek je uporabljena kar največja velikost slovarja $N := 171,476$, ter stopnja uspeha $\pi = 0.99$, kar pomeni, da napadalec lahko dešifrira skoraj vsa sporočila. Iz teh parametrov lahko izračunamo število poskusov, ki jih mora narediti napadalec za vsak e-poštni predal. Pri Apache testnem predalu pridemo do $4.17 \cdot 10^{12}$ poskusov, da napadalec lahko odpre sporočilo. Če to pomnožimo s številom sporočil v predalu in prej omenjeno hitrostjo iskanja in dešifriranja dobimo čas v letih, ki je prikazan v tabeli 5. Z optimizacijo se lahko napad pospeši, tako da se namesto programskega jezika python uporabi programski jezik C. Seveda pa napadalec lahko iz nekaj dešifiranih sporočil izve tudi ključne besede, ki se navezujejo na naprimer podjetje in si tako pomaga pospešiti dešifriranje. Zato je potrebno predstaviti tudi čas za najslabše primere, kjer je 50% besed iz angleškega slovarja in napadalec ve natanko katere besede pripadajo tem 50%, da lahko izvede hitrejši napad. Iz tabele je razvidno, da za najmanjši e-poštni predal še vedno rabi 808.74 let.

Napadalec se lahko odloči, da bo dešifriral le polovico naključnih sporočil, za še hitrejše dešifriranje, kar pa pomeni, da mogoče ne bo dobil informacij, ki jih išče. Ta napad je prikazan v tretji vrstici tabele 5 za $\pi = 0.99$. Za nadaljevanje predvidevamo, da napadalec pozna besedišče podjetja in zmanjša slovar za 50%, ter dešifrira le polovico naključnih sporočil, kar mu da še večjo prednost. Ta rezultat je prikazan v četrti vrstici.

Nazadnje predvidevamo, da ima napadalec še več znanja o besedišču, zato zmanjšamo število besed vseh možnih besed na $20,000$ in $10,000$. Če ta primera združimo s prej omenjenimi metodami, se potreben čas, da napadalec dešifrira polovico sporočil iz nabiralnika C, drastično zmanjša na 1.92 meseca oziroma 0.16 let. Ampak ta primer napada lahko označimo kot nerealen, saj nabiralnik vsebuje le 83 sporočil in malo verjetno je, da ima napadalec toliko znanja o ključnih besedah. Če vzamemo najslabši primer za nabiralnik B, kjer napadalec za dešifriranje porabi 3.5 let, lahko zaključimo, da je zaščita zasebnosti dobra, tudi v zelo slabih primerih.

π	N	Apache	Work	A	B	C
0.99	171,476	$1.15 \cdot 10^8$	$3.26 \cdot 10^8$	$1.23 \cdot 10^8$	$1.17 \cdot 10^8$	5,373.15
0.50	171,476	$1.73 \cdot 10^8$	49,072.84	18,565.34	17,638.94	808.74
0.99	85,738	$1.44 \cdot 10^8$	40,753.36	15,418.00	14,648.51	671.63
0.50	85,738	$2.17 \cdot 10^8$	6,133.99	2,320.64	2,204.82	101.09
0.99	20,000	$1.83 \cdot 10^8$	517.23	195.68	185.91	8.52
0.50	20,000	27,510.37	77.85	29.45	27.98	1.28
0.99	10,000	22,843.44	64.64	24.46	23.24	1.07
0.50	10,000	3,438.28	9.73	3.68	3.50	0.16

Tabela 5: Čas trajanja napada (v letih) za dešifriranje e-poštnega predala z verjetnostjo π in uporabo slovarja z N besedami [3].

4. OMEJITVE

Pri sami rešitvi je nekaj omejitev zaradi zasnove kriptografske sheme. Ker rešitev išče po šifriranem besedilu, razkrije samo besede s popolnim ujemanjem, zato je ta občutljiva na napake pri črkovanju in ne omogoča razkritje delnih ujemanj oziroma uporabo regularnih izrazov.

Naslednja omejitev se navezuje na napade, saj je rešitev možno napasti s slovarjem oziroma z grobo silo. Ampak to je neizogibna omejitev saj moramo preiskovalcu omogočiti iskanje po ključnih besedah. Kot je bilo prikazano v prejšnjem poglavju je zahtevnost uspešnega napada pri realnih primerih dovolj velika in posledično je varnost dovolj dobra.

5. SORODNA DELA

Kar se tiče ohranjanja zasebnosti pri e-poštnih preiskavah s pomočjo kriptografije, do danes ni bilo nobene rešitve. Na temo forenzične analize e-poštnih sporočil najdemo veliko del, vendar ta delajo na surovih podatkih in ne upoštevajo zasebnosti. Kot je na primer analiza e-pošte s pomočjo podatkovnega rudarjenja [24] [23], ter opravljanje podobne tehnike na e-poštnih zbirkah za identifikacijo izvora spama [25]. Kar pa se tiče zasebnosti, obstajajo članki, ki komentirajo splošno zasebnost pri sami forenzični preiskavi [22] [2] [5]. Agarwal in Rodhain sta zelo dobro prikazala pregled pričakovanj pri zasebnosti, še posebej pri e-pošti in delovnih okoljih [1]. Hou in sodelavci so opisali podobno področje, kjer opisujejo rešitev za ohranitev zasebnosti pri preiskovanju skupnih oddaljenih strežnikov [10]. Vendar se ta zelo razlikujejo od dela, ki je opisan v tem članku. Uporabljajo namreč enkripcijo z javnim ključem, kar da veliko režije pri samem računanju in večje potrebe pri shranjevanju podatkov. Kot drugo pa ta rešitev zahteva interakcijo med preiskovalcem in lastnikom podatkov.

Martin Oliver je predstavil sistem za ohranitev zasebnosti zaposlenih pri brskanju po spletu, ki hkrati omogoča forenzikom sledenje viru zlonamernih zahtev [13]. Omenimo lahko še nekaj podobnih del, ki se navezujejo na kriptografsko rešitev. Obstajajo kriptografske sheme, ki omogočajo šifriranje datotek in tudi iskanje po njih s pomočjo ključnih besed [21] [7] [8] . Lastnik podatkov le te zašifrira in ustvari iskalne žetone, ki omogočajo identifikacijo zašifriranih datotek, ki vsebujejo določene ključne besede. Tisti, ki prejme šifrirane datoteke lahko potem s pomočjo ključnih besed identificira pravo zašifrirano datoteko, vendar je sam ne more odšifrirati. Poslati jo mora namreč nazaj lastniku, da jo ta odšifrira. Tukaj spet vidimo že omenjeno slabost, saj mora uporabnik stalno komunicirati z lastnikom šifriranih datotek.

6. ZAKLJUČEK

V seminarski nalogi smo predstavili nov način za forenzično analizo e-poštnih sporočil, ki sta ga zasnovala nemška raziskovalca Frederik Armknecht in Andreas Dewald [3]. Ta način zaščiti zasebnost lastnikov e-poštnih sporočil in hkrati še vedno omogoča forenzično preiskavo. Implementiran program zašifrira tekst tako, da ga preiskovalec lahko odšifrira le ko se v njem pojavi zadostno število ključnih besed iz preiskovalčevega seznama. Število besed, ki se morajo ujemati se določi preden se vsebina zašifrira. Prav tako se določi črni seznam besed, ki se jih pri ujemanju ignorira. Te besede so naprimer ime podjetja ter vse besede, ki se pojavijo v skoraj vsakem tekstu na primer zaimki in pozdravi. Za večjo varnost mora biti črni seznam besed sestavljen zelo premišljeno.

Implementiran je bil tudi prototip, ki zašifrira e-poštne predale različnih formatov v ppef datoteko. Zaradi praktične uporabe je bil razvit še vtičnik za forenzično orodje Autopsy, kjer uporabnik lahko uvozi in preiskuje omenjene ppef datoteke. Opisana je bila zmogljivost in režija implementacije pri šifriranju ter dešifriranju. Opisana rešitev predstavlja nov in varen način zagotavljanja zasebnosti, ne samo pri e-poštni preiskavi ampak tudi na drugih področjih digitalne forenzike.

7. VIRI IN LITERATURA

[1] Ritu Agarwal and Florence Rodhain. Mine or ours: email privacy expectations, employee attitudes, and perceived work environment characteristics. In *System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on*, pages 2471–2480. IEEE, 2002.

[2] Asou Aminnezhad, Ali Dehghantanha, and Mohd Taufik Abdullah. A survey on privacy issues in digital forensics. *International Journal of Cyber-Security and Digital Forensics (IJCSDF)*, 1(4):311–323, 2012.

[3] Frederik Armknecht and Andreas Dewald. Privacy-preserving email forensics. *Digital Investigation*, 14:S127–S136, 2015.

[4] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. The keccak sha-3 submission. `http://keccak.noekeon.org/`, 2011. [Online; accessed 10.6.2016].

[5] Michael A Caloyannides. *Privacy protection and computer forensics*. Artech House, 2004.

[6] Brian Carrier. Autopsy. `http://www.sleuthkit.org/autopsy/`. [Online; accessed 10.5.2016].

[7] Yan-Cheng Chang and Michael Mitzenmacher. Privacy preserving keyword searches on remote encrypted data. In *Applied Cryptography and Network Security*, pages 442–455. Springer, 2005.

[8] Reza Curtmola, Juan Garay, Seny Kamara, and Rafail Ostrovsky. Searchable symmetric encryption: improved definitions and efficient constructions. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 79–88. ACM, 2006.

[9] Joan Daemen and Vincent Rijmen. *The design of Rijndael: AES-the advanced encryption standard*. Springer Science & Business Media, 2013.

[10] Shuhui Hou, Tetsutaro Uehara, Siu-Ming Yiu, Lucas CK Hui, and KP Chow. Privacy preserving confidential forensic investigation for shared or remote servers. In *Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), 2011 Seventh International Conference on*, pages 378–383. IEEE, 2011.

[11] University Leipzip. Wortlisten. `http://wortschatz.uni-leipzig.de/html/wliste.html`, 2001. [Online; accessed 10.5.2016].

[12] National Institute of Standards and Technology. Fips pub 180-2, secure hash standard. `http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf.`, 2002. [Online; accessed 10.6.2016].

[13] Martin Olivier. Forensics and privacy-enhancing technologies. In *Advances in Digital Forensics*, pages 17–31. Springer, 2005.

[14] Informacijski pooblaščenec. Iskalnik po odločbah in mnenjih vop. `http://www.ip-rs.si`, 2016. [Online; accessed 14.5.2016].

[15] Oxford University Press. The second edition of the 20-volume oxford english dictionary. `http://www.oxforddictionaries.com/words/`, 2016. [Online; accessed 10.5.2016].

[16] Universal declaration of human right. 12. člen. *UN General Assembly*, 1948.

[17] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[18] Ustava Republike Slovenije. 37. člen. *Uradni list*, 47, 2013.

[19] Ustava Republike Slovenije. 46. člen. *Uradni list*, 21, 2013.

[20] Ustava Republike Slovenije. 48. člen. *Uradni list*, 21, 2013.

[21] Dawn Xiaodong Song, David Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. In *Security and Privacy, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on*, pages 44–55. IEEE, 2000.

[22] Patrick Stahlberg, Gerome Miklau, and Brian Neil Levine. Threats to privacy in the forensic analysis of database systems. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 91–102. ACM, 2007.

[23] Salvatore J Stolfo, Germán Creamer, and Shlomo Hershkop. A temporal based forensic analysis of electronic communication. In *Proceedings of the 2006 international conference on Digital government research*, pages 23–24. Digital Government Society of North America, 2006.

[24] Salvatore J Stolfo and Shlomo Hershkop. Email mining toolkit supporting law enforcement forensic analyses. In *Proceedings of the 2005 national conference on Digital government research*, pages 221–222. Digital Government Society of North America, 2005.

[25] Chun Wei, Alan Sprague, Gary Warner, and Anthony Skjellum. Mining spam email to identify common origins for forensic application. In *Proceedings of the 2008 ACM symposium on Applied computing*, pages 1433–1437. ACM, 2008.

[26] Wikipedia. Biclique attack — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/wiki/Biclique_attack`, 2016. [Online; accessed 10.6.2016].

[27] Wikipedia. Block cipher mode of operation — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation`, 2016. [Online; accessed 10.6.2016].

E-mail avtorstvo na podlagi asociativne klasifikacije

Tilen Matkovič
Fakulteta za računalništvo in informatiko
Večna pot, 113
Ljubljana, Slovenija
matkovic.tilen@gmail.com

Primož Črnigoj
Fakulteta za računalništvo in informatiko
Večna pot, 113
Ljubljana, Slovenija
primoz.crnigoj@gmail.com

Sašo Pajntar
Fakulteta za računalništvo in informatiko
Večna pot, 113
Ljubljana, Slovenija
sp8867@student.uni-lj.si

POVZETEK

Komunikacija prek elektronske pošte se dostikrat uporablja kot orožje nepridipravov, ki želijo na vsak način izkoristiti socialni inženiring in škodovati ljudem. Protokol za elektronska sporočila ima ta problem, da se lahko kdorkoli predstavlja za neko osebo in vseprek pošilja neka sporočila. Tudi, če izbriše sledi iz glave elektronske pošte ali celo uporabi več računov, lahko, misleč, da ne pusti nobenih sledi, iz sloga pisanja, uporabljanja določenih fraz, ločil in strukture besedila s pomočjo asociativne klasifikacije zmanjšamo število osumljencev in ugotovimo katera oseba je napisala neko elektronsko sporočilo. Na podlagi značilk pisanja in asociativnih pravil zgradimo klasifikator, ki je v tem članku tudi opisan in opisana je tudi sama uporaba le-tega na realnih podatkih.

Ključne besede

elektronsko sporočilo, avtorstvo, podatkovno rudarjenje, asociativna pravila, asociativna klasifikacija

1. UVOD

Elektronska sporočila so eden izmed najbolj pogostih načinov elektronske komunikacije. Pogosto se uporabljajo tudi za pošiljanje neželjenih sporočil, izvajanje napada z ribarjenjem, pošiljanjem zlonamernega programja in druga kazniva dejanja, zato so pogosto predmet preiskav digitalnih forenzikov. Protokol za pošiljanje elektronskih sporočil je varnostno pomankljiv, saj omogoča ponarejanje metapodatkov elektronskega sporočila (angl. *email spoofing*), zato analiza metapodatkov ni primerna za določanje avtorstva. Za dosego tega cilja se vsebina elektronskega sporočila analizira s pomočjo stilometričnih značilnosti, na podlagi katerih se izdela t.i digitalni prstni odtis vsebine (angl. *writeprint*). Digitalni prstni odtis vsebine (podobno kot klasičen prstni odtis) služi temu, da se na podlagi stilometričnih lastnosti pokaže, kdo je avtor elektronskega sporočila.

Stilometrija je veda, ki v besedilih išče vzorce za enolično identifikacijo njihovega avtorja. Cilj je, da se s pomočjo stilometrije odkrije značilnosti pisanja, ki se bodo ponavljale znotraj besedil istega avtorja in na podlagi katerih se bodo avtorji med seboj opazljivo razlikovali [9]. Pri analizi besedila se išče vzorce za naslednje značilnosti:

- leksikografske značilnosti (število črk, število besed, povprečna dolžina besede, povprečna dolžina stavka glede na število črk, itd.),
- sintaktične značilnosti (število ločil, razmerje uporabe ločil glede na uporabo besed, itn.),
- strukturne značilnosti (število vseh vrstic, število praznih vrstic, število stavkov, število odstavkov, povprečna dolžina odstavkov glede na število znakov, besed, stavkov...) in
- vsebinske značilnosti.

Če se pri izdelavi digitalnega prstnega odtisa vsebine upošteva kombinacija značilnosti, potem je rezultat bistveno boljši, kot če se upošteva samo posamezna značilnost. Zato je pomembno, da pri natančnosti analizirnih metod upoštevamo soodvisnost in asociativnost stilometričnih značilnosti. To dosežemo z uporabo metode podatkovnega rudarjenja imenovano asociativna klasifikacija [8].

Asociativna klasifikacija je metoda podatkovnega rudarjenja, ki pri izdelavi modela odločanja uporablja tehnike odkrivanja povezovalnih pravil (angl. *association rule discovery*) in klasifikacije (angl. *classification*). Obe tehniki sta si podobni, le da se odkrivanje povezovalnih pravil osredotoča na povezave med atributi pravil, klasifikacija pa na uvrščanje v razrede glede na podobnost [11]. Obstaja kar nekaj implementacij asociativne klasifikacije:

- klasifikacija glede na asociacijo (angl. *classification based on association* ali CBA),
- klasifikacija glede na napoved asociativnih pravil (angl. *classification based on predictive association rules* ali CPAR),
- klasifikacija glede na napoved več asociativnih pravil (angl. *classification based on multiple association rules* ali CMAR) in

- večrazredna klasifikacija glede na asociativna pravila (angl. *multi-class classification based on association rules* ali MCAR).

Za določanje avtorstva elektronskih sporočil je potrebno natančno izmeriti podobnosti med različnimi stili pisanja napisanih sporočil, zato se je pri analizi uporabila metoda CMAR, saj ta pri določanju najboljšega zadetka med avtorji elektronskega sporočila uporablja podmnožico asociativnih pravil, za razliko od drugih tehnik, ki uporabljajo samo najboljše pravilo. Za razliko od tradicionalnih metod odkrivanja povezovalnih pravil, kjer se išče povezave glede na učno množico, se v tem primeru išče povezave tudi med samimi povezovalnimi pravili. S tem se pridobi kombinacije značilnosti, ki so pomembne pri iskanju razlik med avtorji v procesu iskanja avtorstva. V kombinaciji s pravili, povezanimi z razredi (v našem primeru to predstavlja posamezne avtorje), tak model predstavlja dopusten dokaz o avtorstvu spornega elektronskega sporočila [8].

2. SORODNA DELA

Ugotavljanje avtorstva spada med kategorizacijo besedila in je klasifikacijski problem. Klasifikacijski model je zgrajen na podlagi prejšnjih pisanih dokumentov sumljivih avtorjev. Imena avtorjev so uporabljena kot vrednosti končnega razreda v učni in testni fazi pri razvoju modela. Za razliko od preverjanja avtorstva, ki se obravnava kot eno ali dvorazredni problem, se dodeljevanje avtorstva lahko obravnava kot več-razredni klasifikacijski problem.

Enolična predstavitev atributov, s katerimi bi opisali slog pisanja neke osebe, ne obstaja, vendar nekatere študije so pokazale, da obstajajo določene reprezentativne značilke, ki prispevajo k razvrščanju anonimnih ali spornih besedil. Ločila in n-gramske značilke sta se pokazali za zelo dober pristop, vednar kombinacija obeh skupaj daje še boljše rezultate. Še ena zelo reprezentativna značilka je uporabljanje določenih besed oz. njihovih asociacij namesto nekaterih drugih besed. Bogatost besednjaka, tekoče znanje jezika, slovnične napake, zaporedje besed, struktura besedila, so tudi ene izmed pomembnih značilnosti, s katerimi lahko napovemo avtorja neznanega besedila. Študija [1] razglablja o teh v detajlih.

Večina metod zahteva pravo izbiro atributov, ki privede k večji natančnosti modela; algoritem opisan v tem članku tega ne zahteva, ker nepomembni atributi naj ne bi dosegli minimalnega praga podpore. Z drugimi besedami, algoritem sam izvede izbiro atributov, ki velja za kompleksnejši del naloge.

Analiza avtorstva je bila dokaj uspešna v ugotavljanju avtorstva pri različnih vrstah pisanja. Tovrstno ugotavljanje pri elektronskih sporočilih pa je toliko težje. Za razliko od literarnih del, so elektronska sporočila krajša in ponavadi nimajo prave oblike in slovničnih pravil. Zaradi tega je težje najti vzorce, ki bi kazali na nekega avtorja. Ugotavljanje avtorstva naj ne bi bilo dovolj natančno za besedila, katera vsebujejo manj kot 500 besed [6]. Kakorkoli, elektronska sporočila so neformalna in za tovrstno komunikacijo ni nujno da bodo brez kakršnihkoli napak.

Ugotavljanje avtorstva elektronskih sporočil je možno rešiti z algoritmi, ki upoštevajo specifične izzive pri analizi avtorstva elektronskih sporočil [5]. Algoritem iz našega članka se od tovrstnih algoritmov razlikuje s tem da vključuje tehnike rudarjenja podatkov in se imenuje asociativna klasifikacija. Za razliko od algoritma *AuthorMiner* [5], ki matematično izračuna in poišče pogoste fraze, ter jih nato filtrira, algoritem, opisan v tem članku, dosega višjo točnost.

Druga metoda, metoda podpornih vektorjev, je bila tudi implementirana za ugotavljanje avtorstva elektronskih sporočil [2]. Izvedeni so bili številni eksperimenti in prišli so do zaključka, da se klasifikacijska točnost zniža, ko se zniža število primerov, zviša število avtorjev in ko se zniža povprečna dolžina dokumentov.

Uspešnost SVM-ja se je zmanjšala tudi, ko se je število besed zvišalo iz 122 na 320, ki pa nasprotuje načelu, da SVM podpira visoke dimenzije in iz tega lahko sklepamo, da višanje števila atributov ne doprinese k večji točnosti. K večji uspešnosti pripomore izbira prave kombinacije atributov, s katerimi lahko razlikujemo med slogi pisanja in izbris neuporabnih atributov [4].

Pokazano je bilo, da lahko dodajanje neuporabnih atributov negativno vpliva na klasifikacijsko točnost, ker klasifikator te atribute zajame kot šum [2]. Podobno, uporabljanje pogostih ali šibkih atributov tudi škoduje. Eden od pristopov opisanih v našem članku premaga to omejitev s fleksibilno ekstrakcijo vsebine in filtriranjem šuma.

3. PROBLEM

Naj bo S skupina anonimnih avtorjev elektronskih sporočil e. Naj bo E_i relativno velika zbirka elektronskih sporočil napisanih z avtorjem $S_i \in S$. Naj bo V niz različnih besed v $\cup E_i$. Problem ugotavljanja avtorjev je iskati najbolj verjetnega avtorja S_a iz S, katerega zbirka elektronskih sporočil E_a najbolj ustreza značilkam zlonamernega elektronskega sporočila e. Zbirka elektronskih sporočil E_i se ujema z e, če si E_i in e delita podobne vzorce pisanja. Glavni cilj kibernetskih forenzičnih preiskovalcev je samodejno in učinkovito modelirati vzorce vsakega osumljenca. Ti vzorci lahko služijo kot dokaz za identifikacijo avtorja zlonamernega elektronskega spročila e.

Iz vzorcev želimo izločiti pravila, ki enolično predstavljajo slog pisanja vsakega osumljenca S_i, vendar ne poosebljajo katere koli druge osebe S_j $(i \neq j)$.

3.1 Izločevanje značilk

Za vsako sporočilo najprej izločimo glavo in ostalo redundantno vsebino. Sporočila z manj kot nekaj stavki ali nerelevantim besedilom niso vsebovani, saj ne bi prinesli dovolj informacije za ugotovitev avtorjevega sloga pisanja.

Numerične vrednosti normaliziramo med 0 in 1 in jih nato diskretiziramo na določene intervale, npr. $[0, 0.25]$, $(0.5, 0.75]$, $(0.75, 1]$. Vsak interval naj vsebuje približno enako število vrednosti. Slog pisanja v neki zbirki sporočil E_i, napisanih z osumljencom S_i, je kombinacija značilk, ki se pogosto pojavijo v zbirki sporočil E_i. Te pogosto ponavljajoče vzorce lahko modeliramo z algoritmi iz podveje podatkovnega rudarjenja, kot je naprimer rudarjenje pogostih vzorcev (angl.

frequent pattern mining [4]).

Slog pisanja osumljenca S_i je torej predstavljen kot niz vzorcev, pridobljen iz sporočil E_i. Vzorce lahko nato uporabimo kot zbirko asociativnih pravil.

3.2 Asociativna klasifikacija digitalnega prstnega odtisa vsebine

Podobno kot se za povezovanje prstnih odtisov in osumljencev uporablja daktiloskopija, se za izdelavo digitalnega prstnega odtisa vsebine poslužujemo metod, ki identificirajo različne stile pisanja avtorjev elektronskih sporočil. Za razliko od daktiloskopije, nam ta metoda ne zagotavlja unikatnega razlikovanja med avtorji, je pa dovolj natančna, da pokaže smiselne razlike med stili posameznih avtorjev, če imamo zadostno količino elektronskih sporočil za posameznega avtorja.

Klasifikacija je proces, pri katerem se na podlagi testnega nabora podatkov izdela klasifikator, ki zna natančno klasificirati posamezen podatek iz testnega nabora (v našem primeru gre za klasifikacijo posameznega elektronskega sporočila iz testnega nabora elektronskih sporočil). Obstaja več različnih implementacij algoritmov za klasifikacijo:

- Naivni Bayes - verjetnostni klasifikator, ki na podlagi vrednosti atributov in Bayesovega teorema o pogojnih vrednostih izračuna verjetnost, da neka entiteta pripada določenemu razredu, pri tem pa se predpostavi, da so atributi med seboj neodvisni.
- Odločitvena drevesa (angl. *decision trees*) - so odločitveni sistemi, ki na vsakem nivoju ponudijo odločitev na podlagi enega atributa, ki najbolj razdeli vhodno množico. Zadnji nivo drevesa predstavlja razred, kateremu entiteta pripada.
- Metoda podpornih vektorjev (angl. *support vector machine* ali SVM) - temelji na ideji, da obstajajo jasne meje med razredi v n-dimenzionalnem prostoru. Vsaka entiteta je predstavljena kot vektor tega prostora, n je število atributov, vrednosti atributov pa so koordinate vektorja. S pomočjo hiperravnine se išče mejo, ki bi bila enakomerno najbolj oddaljena od najbljižjih entitet vseh razredov [8, 9].

Novejši pristop h klasifikaciji je, da se poišče povezave med specifičnim naborom atributov in razredom, katerim pripadajo. Tako lahko na podlagi ponavljajočih atributov znotraj entitet istega razredu sklepamo, da tudi druge entitete, ki imajo enak vzorec, pripadajo temu razredu. V našem primeru so entitete elektronska sporočila, njihovi atributi pa so stilometrične lastnosti. Uporaba asociativne klasifikacije pred klasičnimi algoritmi ima prednost tudi v tem, da je rezultat skupek "če-potem" pravil, ki se jih lažje interpretira.

Naj bo S končna množica razredov, kjer vsak razred predstavlja enega avtorja. Učna množica je množica elektronskih sporočil z oznako, kateremu avtorju pripada $S_i \in S$. Klasifikator je funkcija, ki preslika elektronsko sporočilo v specifičen razred. Asociativna klasifikacija je proces odkrivanja asociativnih pravil razreda (angl. *class association rules* ali CAR), ki povezujejo stilometrične značilnosti z avtorji oz. osumljenci. Vsako pravilo vsebuje dva atributa, s katerima omejimo uporabnost pravila, to sta zaupanje (angl. *confidence*) in podpora (angl. *support*). Definiciji atributov sta naslednji:

- Zaupanje: naj bo asociativno pravilo definirano kot $A \rightarrow B$, kjer je $A \subseteq V$, $B \in S$. Zaupanje $sup(A \rightarrow B)$ je definirano kot procent vseh elektronskih sporočil, ki vsebujejo $A \cup B$.
- Podpora: naj bo asociativno pravilo definirano kot $A \rightarrow B$, kjer je $A \subseteq V$, $B \in S$. Podpora $conf(A \rightarrow B)$ je definirana kot procent vseh elektronskih sporočil, katere vsebujejo B ter hkrati tudi A.

Uporabnost pravila omejimo z minimalnimi vrednostmi zaupanja in podpore ki določajo, ali naj se pravilo uporabi ali ne. Tako je asociativno pravilo razreda definirano kot $sup(A \rightarrow B) \geq min\_sup$ in $conf(A \rightarrow B) \geq min\_conf$, kjer je min_sup minimalna vrednost za podporo in min_conf minimalna vrednost za zaupanje. Minimalne vrednosti so določene zato, da se znebimo t.i šuma - to so napake bodisi zaradi napačne klasifikacije razreda ali napake pri merjenju atributov.

Za primer vzemimo, da 90% elektronskih sporočil S_i osumljenca vsebuje 3 odstavke, potem je zaupanje takega pravila $conf(3\ odstavki \rightarrow S_i)$ 90%. Tako pravilo lahko uporabimo za klasifikacijo drugih elektronskih sporočil, ki vsebujejo isti vzorec, v tem primeru 3 odstavke.

Klasifikacija nove entitete običajno poteka tako, da se pridobi vsa pravila, ki ustrezajo minimalnim vrednostim zaupanja in podpore, nato se izbere ustrezno pravilo z najvišjim zaupanjem in podporo ter se z njo klasificira nova entiteta. Novejši primeri asociativne klasifikacije pravila tudi rangirajo ter režejo (angl. *prune*), ali pa uporabijo več pravil hkrati za klasifikacijo v primeru, kjer klasifikacija po najboljšem pravilu ni najboljša izbira, kar prikazuje naslednji primer. Predpostavimo, da hočemo poiskati avtorja anonimnega elektronskega sporočila z značilnostmi (2, 5, 8). Ustrezna pravila z najboljšo podporo so naslednja:

- Pravilo R1:2 $\rightarrow$ Osumljenec 0 (zaupanje: 33%, podpora: 90%)
- Pravilo R2:5 $\rightarrow$ Osumljenec 1 (zaupanje: 67%, podpora: 89%)
- Pravilo R3:8 $\rightarrow$ Osumljenec 1 (zaupanje: 50%, podpora: 88%)

Večina metod asociativne klasifikacije bi na podlagi pravila z največjo podporo klasificiralo elektronsko sporočilo osumljencu 0, brez upoštevanja ostalih dveh pravil. Podroben pregled pravil nam pokaže, da imajo vsa tri pravila podobno podporo, vendar pa imata pravili R2 in R3 višjo vrednost zaupanja, kar pomeni, da so se te značilnosti v učni množici večkrat pojavile pri osumljencu 1. Tako je osumljenec 1 bolj

intuitivna izbira, sploh v primeru, ko se odloča med osumljenčevo nedolžnostjo ali krivdo. Zato se pri klasifikaciji elektronskih sporočil uporablja več pravil, saj so rezultati bolj točni. Paziti je potrebno, da se ne zavrže preveč informacij, saj se običajno pravila z nižjimi vrednostmi zaupanja in podpore režejo oz. nadomestijo z močnejšimi pravili, ne glede na to, katerim razredom pripadajo. To pomeni, da se lahko določenega avtorja poveže z neznanim sporočilom zgolj zato, ker ima močnejši digitalni prstni odtis vsebine od resničnega avtorja. To je podobna situacija kot primerjanje ujemajočega prstnega odtisa s popolnim in delnim prstnim odtisom. Popolni prstni odtis ima večjo verjetnost ujemanja oz. neujemanja glede na celoto, delni prstni odtis pa se lahko v celoti ujema, vendar se ga izreže, ker gre le za delni prstni odtis in je ujemanje občutno manjše kot pri popolnem prstnem odtisu.

Ko imamo enkrat množico asociativnih pravil razreda, je potrebno izrezati vsa pravila, ki so skupna ostalim osumljencem, saj hočemo določiti osumljenca po njegovih unikatnih stilometričnih značilnostih. Množica pravil se imenuje WPCAR (angl. *writeprint class association rules*) in zajema digitalne prstne odtise vsebin osumljencev. Definicija je naslednja:

- *WPCAR*: digitalni prstni odtis vsebine osumljenca S_i, označen kot *WP(S_i)*, je množica pravil v *WPCAR*, ki imajo obliko $A \rightarrow S_i$.

Izrezovanje skupnih pravil močno poveča kvaliteto digitalnega prstnega odtisa vsebine osumljenca, saj pravila unikatno določajo njegove stilometrične lastnosti. Pomankljivost te metode pa je, da lahko pride do prekrivanja digitalnih prstnih odtisov osumljencev, kar pomeni, da taki osumljenci zaradi izrezovanja skupnih pravil lahko ostanejo brez digitalnega prstnega odtisa vsebine. To se zgodi v primeru, ko za določenega osumljenca veljajo vsa pravila, ki so skupna tudi drugim osumljencem.

Problem klasifikacije po več asociativnih pravilih lahko razdelimo na tri podprobleme:

1. za vsakega osumljenca S_i je potrebno poiskati CAR pravila za digitalni prstni odtis vsebine *WP(S_i)* iz učne mnižice elekronskih sporočil E,
2. identificirati je potrebno avtorja zlonamernega elektronskega sporočila e z iskanjem po *WP(E_1), ... WP(E_n)*,
3. izvleči je potrebno prepričljive dokaze o avtorstvu osumljenca.

Za rešitev problema 1 in 2 se uporablja tehnika podatkovnega rudarjenja, kjer se izdela množica pravil glede na ponavljajoče vzorce ter seznam CAR pravil, ki so bila pridobljena pri analizi učne množice elektronskih sporočil E. Po rangiranju in izrezovanju dobimo množico pravil, katere se uporabi kot dokaz in služijo kot podpora za izdelavo zaključkov glede avtorstva (problem 3) [8].

4. KLASIFIKACIJA GLEDE NA NAPOVED VEČ ASOCIATIVNIH PRAVIL ZA DOLOČANJE AVTORSTVA

V tem razdelku je opisana klasifikacija glede na napoved več asociativnih pravil za določanje avtorstva (angl. *classification based on multiple association rules for authorship attribution* ali CMARAA), kot so jo predlagali v članku [10]. Najprej se izlušči same stilometrične vzorce, nato reže pravila, na koncu pa klasificira novo elektronsko sporočilo glede na dobljene digitalne prstne odtise vsebine.

4.1 Rudarjenje asociativnih pravil

Seznam CAR pridobimo z rudarjenjem učnih podatkov, pri čemer dobimo le pravila, ki so dosegla določeno mejo podpore in zaupanja. To je v splošnem proces večine algoritmov za asociativna pravila. Algoritem za iskanje pravil, uporabljen v našem primeru, je različica *FP-growth*-a [3]. Z uporabo drevesne strukture, najde podporo, pri kateri je minimizirano število iskanj po podatkih. Prednost te metode se pozna pri velikih količinah podatkov z majhno podporo in velikim številom atributov.

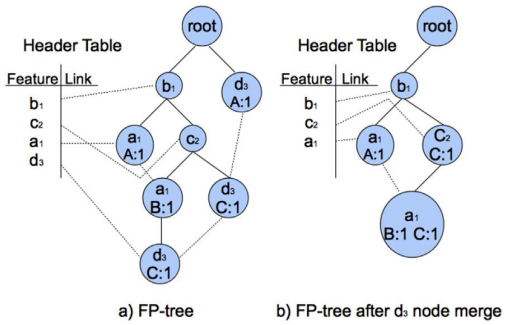
V nadaljevanju je opisan primer delovanja omenjenega algoritma [7].

Recimo da imamo elektronska sporočila z določenimi atributi 1, podporo nastavljeno na 2 in zaupanje na 50%.

Tabela 1: Primer značilk elektronskih sporočil in avtorjev

E-sporočilo	A	B	C	D	Avtor
1	a2	b1	c2	d1	A
2	a1	b1	c2	d3	B
3	a3	b2	c1	d2	A
4	a1	b1	c3	d3	C
5	a1	b1	c2	d3	C

- Najprej se pregleda vsak atribut, kjer izločimo tiste vrednosti, ki ne presegajo minimalnega praga podpore. V našem primeru ostanejo $F = \{a_1, b_1, c_2, d_3\}$.
- F se nato sortira padajoče ($F = \{b_1, a_1, c_2, d_3\}$). *FP-drevo* sestavimo tako, kot je razvidno iz slike 1a. Vsako terko iz F (npr. (b_1, a_1)) vstavimo v drevo, najprej v levo vejo, in ji pripišemo končni razred. Nato nadaljujemo z naslednjo terko (npr. (b_1, c_2, a_1)) in ponovimo proces. V primeru da se poti ponovijo, lahko drevo poenostavimo.
- Pravila iz *F-lista* lahko razdelimo na 4 veje:
  - d_3
  - a_1 brez d_3
  - c_2 brez d_3 ali a_1
  - samo a_1
- Algoritem za iskanje pravil poteka tako, da iz danega drevesa potuje od korena do lista in si zapomne prava vozlišča (npr. $\{b_1, c_2, a_1, d_3\} : C, \{b_1, c_2, d_3\} : C, d_3 : A$). Glede na to, da se d_3 pojavi v vseh pravilih, se iskanje pogostih vzorcev za d_3 lahko poenostavi. Vidimo



Slika 1: Rudarjenje asociativnih pravil [10]

tudi, da se b_1 in c_2 vedno pojavita skupaj. V našem primeru, ko smo pregledali vsa pravila z d_3 vrednostjo, lahko drevo zmanjšamo tako, da vse d_3-je "združimo" v starša (slika 1b).

Celoten postopek se nato ponovi za preostala pravila.

4.2 Rezanje asociativnih pravil

Podatkovno rudarjenje asociativnih pravil ustvari veliko množico pravil, zato je potrebno izrezati podvojena pravila in pravila, ki povzročajo šum. Sledi sortiranje pravil glede na njihov rang. Če imamo dve pravili, recimo *R1* in *R2*, bo pravilo *R1* višje rangirano kot *R2*, če in samo če bo zadostilo naslednjim trem pogojem:

1. *conf(R1)* > *conf(R2)*
2. *conf(R1)* = *conf(R2)*; *sup(R1)* > *sup(R2)*
3. *conf(R1)* = *conf(R2)*; *sup(R1)* = *sup(R2)*; $|ant(R1)| < |r(R2)|$

V prvem krogu se izrežejo pravila, ki so specifična in imajo nižjo stopnjo podpore. Velja namreč, da splošna pravila z višjo stopnjo podpore bolje predstavljajo analizirane značilnosti, kot pa bolj specifična pravila z nižjo stopnjo podpore. Definicija splošnega pravila je naslednja: če imamo dve pravili, *(R1: P → c)* in *(R2: P' → c')*, potem je pravilo *R1* splošno, če in samo če velja $P \subseteq P'$. Takšno izrezovanje (CMAR rangiranje) ne vpliva na točnost analize, ima pa vpliv na njeno učinkovitost. Ker pa se pri analizi elektronskih sporočil išče unikatne stilometrične značilnosti, je potrebno dati prednost specifičnim pravilom. Zato je vpeljana CMARAA. Definicija specifičnega pravila je naslednja: če imamo dve pravili, *(R1: P → c)* in *(R2: P' → c')*, potem je pravilo *R1* specifično, če in samo če velja $P \supseteq P'$. Za razliko od CMAR rangiranja ima CMARAA tretje pravilo rangiranja definirano kot:

3. *conf(R1)* = *conf(R2)*; *sup(R1)* = *sup(R2)*; $|ant(R1)| > |r(R2)|$

Prvi krog izrezovanja na podlagi CMARAA je tako ravno obraten kot pri CMAR, saj se da prednost specifičnim pravilom z visoko podporo pred splošnimi pravili, s tem pa dosežemo večjo točnost pri klasifikaciji. Nadaljne izrezovanje se vrši, ko se pravilo prvič ustavi v drevo klasifikacijskih pravil (angl. *classification rule tree*).

V drugem krogu izrezovanja se uporabi samo pravila, ki so pozitivno soodvisna. Pozitivno soodvisnost se določi na podlagi "chi" kvadrat statističnega testa (angl. *chi square testing*), opravlja pa se pri vnosu v drevo klasifikacijskih pravil.

Tretji krog izrezovanja se vrši na podatkovni bazi, kjer se uporabi t.i prag pokritosti (angl. *database coverage treshold*) zato, da se zmanjša število pravil, ohrani pa se pravila, ki so značilne za učni zapis (angl. *trainig record*). Vhodni podatki so sestavljeni iz seznama pravil in praga pokritosti τ, izhodni podatki pa predstavljajo podmnožico asociativnih pravil. Algoritem se izvaja po naslednjih korakih:

1. sortiranje pravil po rangu,
2. za vsak učni zapis se postavi števec na 0,
3. za vsako pravilo R se poišče vse ujemajoče učne zapise. Če pravilo ustrezno klasificira vsaj en zapis, potem povečamo števec vsem zapisom, ki se ujemajo s pravilom R. Učni zapis se odstrani iz podatkovne baze, ko števec preseže vrednost praga pokritosti τ.

Sama implementacija algoritma je malenkost spremenjena, saj se učnega zapisa ne odstrani takoj, ko je pokrit s katero izmed izbranih pravil, ampak se ga pusti v učni množici do takrat, ko je pokrit z najmanj tremi izbranimi pravili. To vpliva na bolj točno klasifikacijo novih objektov.

4.3 Avtorska klasifikacija

Ko imamo prečiščeno množico asociativnih pravil, se začne proces klasifikacije anonimnih elektronskih sporočil. Če vsa pravila, katera ustrezajo analiziranemu elektronskemu sporočilu pripadajo istemu razredu, potem se elektronsko sporočilo klasificira temu razredu brez nadaljne analize. Če obstajata dve ali več pravil, ki pripadajo različnim razredom, potem naredimo skupine pravil, ki pripadajo posameznemu razredu. Elektronsko sporočilo se klasificira v skupino, ki ima največjo moč glede na vsebovana pravila. Moč skupine je določena na podlagi vsebovanih pravil, saj pravila, ki imajo večjo pozitivno soodvisnost in podporo, dajo tudi večjo moč množici. Algoritmi za asociativno klasifikacijo običajno izberejo za klasifikacijo najmočnejše pravilo, torej pravilo z najvišjim rangom. To je lahko problematično, saj se lahko s tem izbere manj pomembne razrede, kot je prikazano v naslednjem primeru. Imamo dve pravili z definicijo:

- *R1*: značilnost A = ne → avtor B (podpora = 450, zaupanje = 60%)
- *R2*: značilnost B = da → avtor A (podpora = 200, zaupanje = 99.5%)

Izmerjene in pričakovane vrednosti za obe pravili se nahajajo v tabeli 2.

Na podlagi izmerjenih in pričakovanih vrednosti, je vrednost chi faktorja za *R1* 97.6, za *R2* pa 36.5. Tako bi za avtorja

Tabela 2: Izmerjene in pričakovane vrednosti

	avtor A	avtor B	Skupaj
R1 izmerjen			
Značilnost A	410	40	450
Brez značilnosti A	20	30	50
Skupaj	430	70	500
R2 izmerjen			
Značilnost B	209	1	210
Brez značilnosti B	241	49	290
Skupaj	450	50	500
R1 pričakovan			
Značilnost A	387	63	450
Brez značilnosti A	43	7	50
Skupaj	430	70	500
R2 pričakovan			
Značilnost B	189	21	210
Brez značilnosti B	261	29	290
Skupaj	450	50	500

anonimnega elektronskega sporočila, ki ne vsebuje značilnosti A in vsebuje značilnost B predvideli določili avtorja B (pravilo *R1*), če bi se odločali samo med vrednostmi chi faktorja. Če primerjamo zaupanje obeh pravil vidimo, da je pravilo *R2* boljše, saj ima višjo vrednost zaupanja. Kot je razvidno iz tega primera, izbiranje močnejših pravil ni enostavno. Na podlagi testov in meritev [7] se je za izbiro močnejših pravil izbral algoritem chi kvadrat s tehtanjem (angl. *weighted chi square* ali WCS), saj meri moč pravil v razredih na podlagi njihovih soodvisnosti in popularnost. Za vsako pravilo $R{:}P \rightarrow c$ velja, da je *sup(c)* število zapisov v učni množici, ki so povezani z razredom c, ter da je |T| število vseh zapisov v učni množici. Največja vrednost za chi je tako definirana po naslednji formuli:

$$max_{\chi^2} = (min(sup(P), sup(c)) - \frac{sup(P)sup(c)}{|T|})^2|T|e \quad (1)$$

Vrednost e je definirana kot:

$$e = \frac{1}{sup(P)sup(c)} + \frac{1}{sup(P)(|T| - sup(c))} + \frac{1}{|T| - sup(P)sup(c)} + \frac{1}{(|T| - sup(P))(|T| - sup(c))} \quad (2)$$

Za vsako skupino pravil se vrednost izračuna po naslednji formuli:

$$\sum \frac{(\chi^2)^2}{max(\chi^2)} \quad (3)$$

Skupine asociativnih pravil so intuitivno definirane. Naj bo A skupina osumljencev in naj bo F_i značilnost v množici značilnosti F, ki jih algoritem uporablja. Skupina asociativnih pravil, ki je definirana po treh značilnostih, je definirana kot:

$F_1 + F_6 + F_{90} \rightarrow A$
$F_5 + F_{124} \rightarrow A$
$F_{45} + F_{89} + F_{94} + F_{213} \rightarrow A$

5. POSKUSNO VREDNOTENJE

Da bi ocenili natančnost, učinkovitost in skalabilnost algoritma CMAR in predlagane implementacije tega algoritma za pripisovanje avtorja, smo združili rezultate celovite študije. V tem poglavju primerjamo CMAR in CMARAA z dvema znanima klasifikacijskima metodama: klasifikacija z asociacijo in *AuthorMiner*, do sedaj vodilnim algoritmom za ugotavljanje avtorstva. Dodatno CMARAA evalviramo tudi glede na bolj znane klasifikatorje vključujoč naivnim Bayesom (angl. *Naive Bayes*), Bayesovskimi mrežami (angl. *Bayesian Networks* - BayesNet), ansambli ugnezdenih dihotomij (angl. *Ensemble of Nested Dichotomies* - END) in odločitvenimi drevesi (angl. *Decision Trees*), ki se pogosto uporabljajo v študijah za ugotavljanje avtorstva. Izkaže se, da ima CMARAA ravno tako dobro ali še boljšo povprečno natančnost, kot druge metode vključno z CBA in AM. Klasifikacijske tehnike z naključnimi gozdovi, SMO in SVM niso vključene v študijo, saj jih je težko interpretirati, cilj te študije pa je inituitivno predstaviti rezultate.

Poskusno vrednotenje je bilo izvedeno s strani avtorjev članka [10]. Vsi testi so bili izvedeni na 3.4GHz Core i7 CPE z 12GB delovnega pomnilnika, z operacijskim sistemom Mac OS 10.7.3. CMAR in CBA je implementiral Frans Coenen, ko je demonstriral moč in skalabilnost metode Apriori-TFP. AuthorMiner pa je bil implementiran s strani njegovih avtorjev.

Uporabljena je bila Enron zbirka elektronskih sporočil, ki je tudi najbolj primerna javna zbirka, po besedah B.Allison-a in L.Gutheri-a. Natančneje, izbrali smo na stotine sporočil avtorjev: Paul Allen, John Arnold, Sally Beck, John Dasovich, Mark Headicke, Vince Kaminsky, Steven Kean, Kam Keiser, Philip Love, Kay Mann, Susan Scott, Carol St Clair, Kate Symes in Kim Watson. V zbirki sicer nastopa več avtorjev, vendar imajo navedeni največ sporočil. Glave in vsa vsebina sporočil, ki ni bila napisana s strani avtorja sporočila, je bila očiščena. S tem je zagotovljeno, da smo obdelovali zgolj tekst, ki ga je napisal avtor sporočila. Ostalo je samo približno 20% vsebine, ostalo so bili krajši sestavki, posredovane priponke in prav tako tudi testna sporočila. Ne bi bilo smiselno pričakovati natančne klasifikacije kratkih sporočil, sestavljenih iz zgolj nekaj besed. Kratka sporočila bi bilo trivialno prirediti tako, da ne vsebujejo prepoznavnega stila pisanja. Stil bi lahko celo ponaredili, če bi bil ponarejevalec dobro seznanjen z metodologijami za odkrivanje avtorstva.

Vsi rezultati so povprečja natančnosti izračunana iz rezultatov algoritmov nad množico podatkov z istim številom avtorjem, vendar drugačno kombinacijo le teh. S tem pokažemo, da so rezultati stabilni in ne zgolj ugibanja ali namensko izbirne množice. Po pričakovanjih, se pri množici avtorjev z nižjo natančnostjo le-ta pokaže pri vseh algoritmih. Vsak test je pošten in testne množice niso bile priprejene v prid nobenemu algoritmu.

Vsi rezultati upštevajo delež pravilnega ujemanja avtorjev. Na primer, če v 4ih primerih pravilno ugotovimo avtorja v

3 primerih, potem je delež 75%. Testi so bili izoblikovani tako, da je bila celotna zbirka razdeljena v učne in testne množice, pri čemer učne množiče predstavljajo 90% vseh podatkov, testne množice pa 10%. To pomeni, da od 1000 sporočil 900 sporočil vzamemo za učenje modela, 100 sporočil pa za testiranje. Pri algoritmih AuthorMiner in AC se učna množica uporabi za odkrivanje pogostih vzorcev. Na podlagi odkritih vzorcev pa klasificiramo elektronska sporočila iz testne množice. Delitev sporočil v testno in učno množico je narejena na podlagi posamenih avtorjev. Vsak avtor je torej zastopan v učni in v testni množici. Pri vseh algoritmih uporabljamo enak postopek za ustvarjanje testne in učne množice. S tem zagotavljamo da so rezultati direktno primerljivi.

Ponovljivost testov dosežemo tako, da množico podatkov najprej uredimo po abecednem vrstnem redu. Glede na željeno razmerje med učno in testno množico, vedno najprej vzamemo delež za učno množico in nato delež za testno množico. To lahko povzroči različno težavnost klasifikacije sporočil, ki so na začetku ali pa na koncu množice. Naloga vsakega algoritma je, da to težavo uspešno obvlada.

Naslednji del poglavja opisuje rezultate različnih testov. Pri poskusih so bili uporabljeni naslednji parametri. Pri algoritmih CBA in CMAR nastavimo podporno mejo na 10% in stopnjo zaupanja na 0%. S takšno stopnjo zaupanja demonstriramo učinek zadnjega rezanja na natančnost pri algoritmu CMARAA. Visoka stopnja zaupanja bi izločila skupna pravila različnih avtorjev. Nastavljanje različne stopnje zaupanja pri različnih algoritmih bi onemogočilo pošteno primerjavo. Ker stopnja zaupanja pri algoritmih CMAR, CBA in CMARAA ne vpliva na natančost, smo sklepali da je varno in pošteno, če jo nastavimo na 0%. Glavna funkcija stopnje zaupanja in podporne meje je izboljšava hitrosti in ne natančnosti.

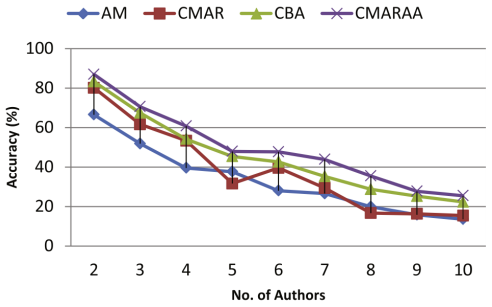
CMARAA uporablja enako stopnjo zaupanja in podporno mejo kot CBA in CMAR, vendar je podporna meja določena posebej za vsakega avtorja, namesto da bi jo določili za celotno učno množico. S tem dosežemo boljšo razpoznavo pogostih vzorcev glede na posameznega avtorja. To je pomembno, saj bi v nasprotnem primeru CMARAA prepoznaval pogoste vzorce na učni množici in ne pogostih vzorcev za posameznega avtorja. Brez tega dodatka bi bilo težje pripisati sporočilo posameznemu avtorju, kot sedaj prikazujejo rezultati.

Tabela 3 in slika 2 prikazujeta povprečno klasifikacijsko točnost pri zbirkah elektronskih sporočil z od 2 do 10 avtorji za vsak algoritem, ki je bil testiran v tej študiji. Natančnost se razpenja od 30 pa do 92%, pri čemer je najnatančnejši CMARAA, sledi CBA nato CMAR in nazadnje AuthorMiner. Natančnost algoritmov pada s povečevanjem števila avtorjev. Pri CMAR opazimo največji padec natančnosti pri 10 avtorjih. Težavnost klasifikacije je bolj odvisna od unikatnosti avtorjevega stila kot pa na moč algoritmov. V kolikor imata dva avtorja podoben stil pisanja, ju bo težje ločiti, ne glede na to, kateri algoritem uporabimo.

S konstatnim minimalnim pragom podpore CMAR ne najde novih pravil z vsakim tekom, kadar je število avtorjev veliko. Vzrok tega je v dejstvu, da so stili pisanja avtorjev

Tabela 3: Točnost glede na število avtorjev [10]

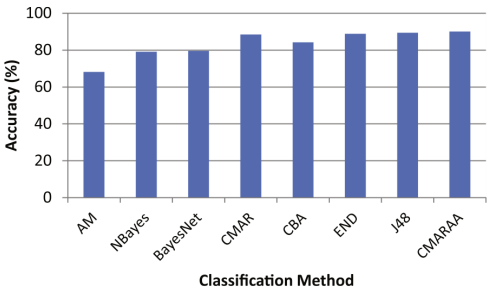
Število avtorjev	AM	CMAR	CBA	CMARAA
2	66.489	80.106	83.229	86.923
3	51.83	61.63	67.37	70.54
4	39.52	53.33	54.12	60.73
5	37.602	31.636	45.34	47.9
6	28	39.535	42.555	47.6925
7	26.5275	29.37	35.2075	43.8675
8	19.8425	16.6633	28.705	25.44
10	15.78	16.32	25.2166	27.69



Slika 2: Točnost glede na število avtorjev [10]

različni in podpora različnim značilnostim ne bo vedno presegla minimalnega praga podpore, ki je odstotek elementov učne množice. Če je minimalni podporni prag 10% in imamo 10 avtorjev, vsakega z 100 sporočili, potem bi se morala značilnost nekega avtorja pokazati v vseh njegovih sporočilih, da bi jo algoritem zaznal kot pogosto značilnost. Eden glavnih doprinosov te študije, implementirane v CMARAA, zniža podporni prag, da se upošteva le sporočila posameznega avtorja. Na ta način lahko izluščimo pogoste značilnosti na nivoju avtorja.

Nazadnje primerjamo predlagano metodo z znanimi klasifikatorji, ki se jih uporablja za določanje avtorstva. Slika 3 in tabela 4 predstavljata povprečne rezultate algoritmov AuthorMiner, naivni Bayes, Bayesovske mreže, CMAR, CBA, END, odločitvena drevesa in CMARAA. Cilj testa je razlikovati med dvema avtorjema v 6 poskusih različnih parov avtorjev. Pokaže se isti trend kot na sliki 2, CMARAA je enako dober ali boljši od ostalih metod.



Slika 3: Točnost klasifikacijskih metod [10]

Tabela 4: Točnost klasifikacijskih metod [10]

Podatki	AM	NBayes	BayesNet	CMAR	CBA	CMARAA
2a	65	75	75	100	100	100
2b	57	85	85	80.18	92.79	83.78
2c	75	76	76.47	87.5	87.5	87.5
2d	66	76.31	76.31	81.89	81.08	83.78
2e	64.86	78.51	80.74	83.33	86.11	91.66
2f	81.25	83.64	83.85	97.92	57.88	93.75
Povprečje	68.19	79.08	79.56	88.47	84.18	90.08

Podrobnih poročil o uporabi CPE in V/I avtor članka [10] ne predstavi. Predpostavlja, da v kontekstu kriminalistične preiskave čas izvajanja v primerjavi s kvaliteto klasifikacije ne igra velike vloge.

Tabela 5 prikazuje povprečen čas izvajanja vsakega algoritma v sekundah, ki so jih algoritmi potrebovali za klasifikacijo množic z 2, 3, 4 in 5 avtorji. Algoritmi, ki so potrebovali najmanj CPE časa za 2 avtorja, so bili: CBA, CMAR, CMARAA in AM. Za 3 avtorje je bil prvi CMAR, sledili so mu CBA, CMARAA in AM. Enak vrsti red lahko opazimo tudi pri 4 avtorjih. Pri 5 avtorjih je bil prvi CBA, nato CMAR, CMARAA in AM. Na tabeli 5 so prikazani časi primarno rezultat njihovih pripadajočih procesov za odkrivanje pogostih elementov. AuthorMiner uporablja original Apriori algoritem za odkrivanje pogostih elementov, CMAR, CBA in CMARAA pa uporabljajo veliko hitrejšo različico FP-Growth algoritma, ki uporablja učinkovite drevesne strukture za predstavitev pogostih elementov in razrednih pravil. AM bi lahko izboljšal učinkovitost z uporabo FP-Growth metode, vendar to ni cilj te študije.

Tabela 5: Povprečen čas izvajanja [s] [10]

Št. avtorjev	AM	CMAR	CBA	CMARAA
2	15.717	3.132	3.0599	8.1525
3	52.83	7.27	7.83	55.39
4	575.73	8.44	8.51	96.34
5	443.467	13.24	12.9844	108.7104

Ti rezultati prikazujejo zanesljiv in ponovljiv dokaz, da so metode za pripisovanje avtorstva lahko zelo uporabne. Dokazujejo tudi, da je lahko stil pisanja avtorja modeliran s prepoznavo vzorcev iz transformirane semantične vsebine. Tako dobimo prepoznavne digitalne prstne odtise vsebine.

6. ZAKLJUČEK

Asociativna klasifikacija je metoda podatkovnega rudarjenja, ki se je izkazala kot zelo učinkovita pri določanju avtorstva elektronskih sporočil. Z uporabo asociativne klasifikacije, ki temelji na razredih ter spremenjenim sistemom rezanja in rangiranja pravil, se pridobi bolj natančen digitalni prstni odtis vsebine. Z odstranitvijo značilnosti, ki so skupne različnim avtorjem, pridobimo digitalni prstni odtis vsebine, ki vsebuje unikatne značilnosti avtorja, kateremu pripada. Zaradi intiutivno definiranih pravil ter njihove predstavitve, je digitalni prstni odtis vsebine prepričljiv dokaz na sodišču, ki poleg drugih dokazov odločilno vpliva na odločitev sodišča o osumljenčevi krivdi ali nedolžnosti.

7. VIRI

[1] A. Abbasi and H. Chen. Writeprints: A stylometric approach to identity-level identification and similarity detection in cyberspace. *ACM Transactions on Information Systems (TOIS)*, 26(2):7, 2008.

[2] M. Corney, O. De Vel, A. Anderson, and G. Mohay. Gender-preferential text mining of e-mail discourse. In *Computer Security Applications Conference, 2002. Proceedings. 18th Annual*, pages 282–289. IEEE, 2002.

[3] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *ACM Sigmod Record*, volume 29, pages 1–12. ACM, 2000.

[4] F. Iqbal, H. Binsalleeh, B. C. Fung, and M. Debbabi. Mining writeprints from anonymous e-mails for forensic investigation. *digital investigation*, 7(1):56–64, 2010.

[5] F. Iqbal, H. Binsalleeh, B. C. Fung, and M. Debbabi. A unified data mining solution for authorship analysis in anonymous textual communications. *Information Sciences*, 231:98–112, 2013.

[6] G. Ledger and T. Merriam. Shakespeare, fletcher, and the two noble kinsmen. *Literary and Linguistic Computing*, 9(3):235–248, 1994.

[7] W. Li, J. Han, and J. Pei. Cmar: Accurate and efficient classification based on multiple class-association rules. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 369–376. IEEE, 2001.

[8] B. C. F. Michael R. Schmid, Farkhund Iqbal. E-mail authorship attribution using customized associative classification. *Elsevier*, 14(1):116–126, Avgust 2015.

[9] I. Panker. Avtomatsko določanje avtorstva slovenskih leposlovnih besedil. Master's thesis, Fakulteta za računalništov in informatiko, Ljubljana, 2012.

[10] M. R. Schmid, F. Iqbal, and B. C. Fung. E-mail authorship attribution using customized associative classification. *Digital Investigation*, 14:S116–S126, 2015.

[11] F. A. Thabtah. A review of associative classification mining. *Knowledge Engineering Review*, 22(1):37–65, Marec 2007.

Forenzična analiza digitalnih artefaktov

Blaž Dolenc
Fakulteta za računalništvo in informatiko
Univerza v Ljubljani
dolenc.blaz@gmail.com

Lovro Podgoršek
Fakulteta za računalništvo in informatiko
Univerza v Ljubljani
lp7602@student.uni-lj.si

Luka Košenina
Fakulteta za računalništvo in informatiko
Univerza v Ljubljani
lk9027@student.uni-lj.si

POVZETEK

Forenzična analiza artefaktov storitev v oblaku je še vedno v začetnih povojih. Trenutni pristopi vse preveč sledijo tradicionalnim metodologijam pridobivanja artefaktov na strani klienta. V tem delu, predstavljamo koncept analize artefaktov, ki so na strani storitev v oblaku, to so podatki, ki hranijo aktualno stanje spletnih/SaaS aplikacij. Delovanje spletnih aplikacij se povsem razlikuje od delovanja tradicionalnih, spletne aplikacije med delovanjem prenesejo le nujno potrebno stanje datotek ter ob končanem delu za seboj ne puščajo sledi. Medtem, ko namizne aplikacije hranijo trenutno stanje datotek na lokalnem datotečnem sistemu.

Z uporabo Google Docs spletnih aplikacij predstavljamo povsem drugačno strukturo artefaktov, kot smo jo navajeni na namiznih aplikacijah. Stanje le teh je navadno v obliki popolnih ali delnih zabeležk uporabnikovih dejanj urejanja dokumenta. Zato je tradicionalni pristop zajema stanja artefaktov v času samo po sebi forenzično pomanjkljivo saj ne upošteva potencialno pomembnih informacij o razvoju dokumenta v daljšem časovnem obdobju. Artefakti na strani strežnika nimajo standardizirane oblike, kar vzbuja vprašanja v zvezi z njihovo dolgoročno ohranitvijo in interpretacijo.

Ključne besede

Forenzika storitev v oblaku, SaaS, Google Docs

1. UVOD

Tradicionalni poslovni model programske industrije je bil tako imenovani *software as a product* (SaaP). To pomeni, da je uporabnik kupil programsko opremo kot fizični produkt, in enkrat ko je prodaja zaključena, kupec lahko uporablja izdelek za nedoločen čas. Alternativni pristop je, tako imenovani *software as a service* (SaaS), kjer že iz imena lahko razberemo, da gre za naročniško orientirani poslovni model. Konceptualno, prehod iz SaaP na SaaS prenese odgovornost uporabe programske opreme in njenega okolja iz kupca na ponudnika storitve. Tehnološko gledano je bil takšen preskok posledica rasti popularnosti interneta in razvoj spletnega brskalnika.

Tradicionalni analitični model digitalne forenzike je bil uporabniško orientiran, kjer je preiskovalec delal z fizičnimi mediji, kot so prenosni mediji ali integrirane računalniške naprave (npr.: pametni telefoni). Na napravi uporabnika je preprosto identificirati kje se izvajajo računske operacije in kje se hranijo rezultati ali pa sledil le teh. Zato so raziskave bile osredotočene na odkrivanje in pridobivanje vsak majhen košček zabeležke ali vsak bit zavrženih podatkov s strani aplikacij oz. operacijskega sistema.

Prihod spletne storitve Gmail leta 2004 je demonstriral stik vseh potrebnih tehnoloških predpogojev za masovno uporabo spletno usmerjenih aplikacij. Leta 2006 je sledila predstavitev prve javne oblačne storitve s strani podjetja Amazon, ki je omogočila vsakemu posamezniku najem razširljive, strežniške infrastrukture. Desetletje za tem je prehod namiznih aplikacij na splet v polnem zagonu, posledično postaja čedalje bolj kritična tudi potreba po obravnavanju tovrstnih storitev na področju digitalne forenzike.

Tovrstni masivni tehnološki preskok predstavlja nov izziv za forenzike, katerega se ne moramo lotiti le s preprostim prilagoditvami forenzičnih orodij in postopkov. Natančneje, SaaS model posega v vsem znani uporabniško osredotočen svet, tako programska koda kot podatki se sedaj dostavljajo preko interneta na zahtevo. Forenzični cilji so tako postali premikajoče se tarče. Na primer, Google Docs dokument je na lokalnem disku predstavljen kot internetna povezava, dejanska vsebina se prenese le ob urejanju v spletnem brskalniku.

Pristop raziskave prikazuje neposreden dostop do vira podatkov spletnih aplikacij z uporabo javnih in zasebnih API storitev. Ta način predstavlja predogled, kako bodo v prihodnje izdelana forenzična orodja.

2. S TEM POVEZANO DELO

Predhodna dela na področju forenzike v oblačnih storitvah so se posluževala tradicionalnega forenzičnega pristopa, iskanja dokazov na strani uporabnika. To obsega tako imenovano blackbox diferencialno analizo, kjer pred in po ustvaritvijo slik izvedemo primerjavo ključnih funkcij aplikacije. Spodnji razdelek (Uporabniško osredotočeno pridobivanje podatkov in njihova analiza) povzema relevantno delo na tem področju. Razdelek (API osredotočeno pridobivanje po-

datkov in njihova analiza) predstavlja bolj sodobno alternativo, ki si prizadeva preprečiti omejitve forenzične obravnave strank s pomočjo ponudnikovih API storitev.

2.1 Uporabniško osredotočeno pridobivanje podatkov in njihova analiza

Chung (2012) je analiziral štiri ponudnike oblačnih storitev (Amazon S3, Google Docs, Dropbox in Evernote). V raziskavi je iskal sledi aplikacij na strani uporabnika, se pravi na napravi s katero je uporabnik dostopal do storitev na oblaku. V poročilu je bilo navedeno, da analizirane storitve ustvarjajo različne artefakte glede na specifične funkcionalnosti storitve. Predlagan je bil procesni model forenzične preiskave spletnih storitev glede na zbiranje in analizo artefaktov ciljnih oblačnih storitev iz odjemalčevih naprav. Pristop vključuje zbiranje nestanovitnih podatkov iz Mac/Windows sistemov, za tem pa pridobivanje podatkov iz zgodovine brskanja po internetu, zabeležk in datotek. Na Android mobilnih naprava so za namen pridobivanja podatkov odklenili skrbniški način uporabe, med tem ko so na napravah iPhone uporabili varnostne kopije programa iTunes. Cilj je bil iskanje sledi uporabe spletnih storitev na omenjenih napravah.

V raziskavi Hale (2013) je bil analiziran Amazon Cloud Drive. Obravnavano je bilo tudi ali obstajajo kakršnekoli sledi o dostopu do Amazon Cloud Drive uporabniškega računa. Obstajata dva načina dostopa do Amazon Cloud Drive uporabniškega računa. Prva je preko spletne aplikacije z uporabo spletnega brskalnika. Druga je pa preko odjemalčeve aplikacije, ki jo ponuja Amazon in jo je možno namestiti na sistem. Po analizi obeh metod, so bili najdeni dokazi v zgodovini spletnega brskalnika in v datotekah, ki jih brskalnik uporablja za pred pomnjenje. Prav tako so bile najdeni artefakti v Windows registrih, na privzeti lokaciji namestitve aplikacije in v SQLite podatkovni bazi aplikacije.

Quick and Choo (2013) sta analizirala Dropbox in razpravljala o artefaktih, ki jih ustvari namizni program Dropbox ob dostopu do spletne storitve. Z uporabo zgoščene (hash) analize in iskanja ključnih besed sta poskušala določiti ali je bil uporabljen namizni program Dropbox. To vključuje ekstrakcijo uporabniškega imena iz zgodovine spletnega brskalnika (Mozilla Firefox, Google Chrome in Microsoft Internet Explorer), pregled internetnih povezav do datotek, registrov, raznih datotečnih direktorijev. V nadaljnjem delu sta Quick in Choo uporabila podoben pristop k analizi artefaktov aplikacije Google Drive na strani uporabnikove naprave.

Martini in Choo (2013) sta raziskovala operacije storitve ownCloud, ki je samo gostovana rešitev za sinhronizacijo in delitev datotek. Tukaj je bil pristop rahlo drugačen, saj gre za drugačno nišo, kjer je bolj verjetno, da sta klient in strežnik pod nadzorom iste osebe oziroma organizacije. Uspela sta obnoviti artefakte, vključno z metapodatki o sinhronizaciji in upravljanju podatkov, pred pomnjenjih datotek, ki opisujejo uporabnikove datoteke shranjene na strani klienta ali prenesene v oblačno storitev.

2.2 API osredotočeno pridobivanje podatkov in njihova analiza

Do sedaj obravnavani forenzični pristopi imajo skupno eno veliko predpostavko; da so lahko vsi relevantni podatkovni artefakti zaseženi na strani klienta. Problem je v tem, da to ne drži vedno. Kot prikazuje spodnja slika 1, klienta ne smemo obravnavati kot izvirni vir podatkov. Precej bolj verjetno je, da klient hrani na lastnem sistemu pred pomnjeno različico dokumenta, ki je navadno nepopolna in zastarela oblika dokumenta.

Glede na zgornjo funkcionalno arhitekturo, obstajajo tri velike pomanjkljivosti na področju zajemu spletno gostujočih podatkov na strani klienta:

Delna replikacija. Najbolj očiten problem je ta, da noben od klientov, ki imajo dostop do gostujočega dokumenta ne hrani lokalno popolne kopije dokumenta. Trenutno ponudniki storitev v oblaku ponujajo selektivno replikacijo. Na ta način omogočajo napravam z manj pomnilnika (pametni telefoni) nemoteno delovanje. Nadalje, medtem ko na storitvah v oblaku hranimo čedalje večjo količino podatkov bi postalo nepraktično, kot tudi nepotrebno ohranjati popolno lokalno kopijo. Amazon že ponuja neomejeno skladiščenje podatkov za ceno $60 na leto, kar lahko predstavlja veliko količino podatkov za lokalno hranjenje. Iz forenzičnega stališča klient ne predstavlja popolnega vira za iskanje podatkovnih dokazov.

Različice artefaktov. Večina ponudnikov skladiščnih storitev ponuja avtomatsko sledenje revizijam in hranijo kopije prejšnjih različic uporabnikovih datotek. Kakorkoli, ti niso prisotni v pred pomnilniških datotekah klienta in so le priklicana na zahtevo (preko spletnega vmesnika). Forenzično, je to povsem druga dimenzija preko katere je zajem podatkov na strani klienta slepo in nepopolno.

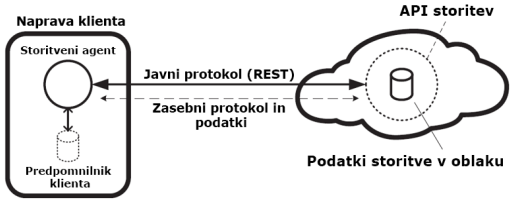


Figure 1: SaaS arhitektura.

Artefakti storitev v oblaku. Spletne aplikacije redko hranijo aktualno stanje na napravah klientov. Obstaja nekaj opaznih izjem, kot so predpomnjenje podatkov o uporabniškem računu in izvajanje operacij v offline delovanju, ampak norma je še vedno, da so vsi podatki gostujoči na strani strežnika. To povzroča vzpon koncepta o artefaktih storitev v oblaku, ki jih uporabljamo za opis notranje strukture uporabljene s strani SaaS aplikacij, ki niso vztrajno lokalno shranjene na strani klienta. To povzroča problem metodam preiskovanja, ki so osredotočene na lokalno stran klienta.

Avtor članka Roussev (2016) trdi, da je edini način s katerim popolnoma naslovi prva dva aspekta tega problema, uporaba uradnih API funkcionalnosti ponudnika spletnih storitev. Izdelali smo orodje imenovano kumodd za zajem podatkov s spletnih storitev, kot so: Google Drive, Dropbox, Box

in Microsoft OneDrive. Orodje lahko našteje in prenese vse datoteke iz enega od pred tem omenjenih storitev. Zajame lahko tudi posnetke v času v standardnih oblikah, kot je na primer PDF.

Pomanjkljivost orodja je nezmožnost zajema tako imenovanih cloud-native artefaktov, saj API funkcionalnosti tega ne omogočajo. Na primer, Google Docs dokument je predstavljen kot spletna povezava in API ne omogoča na takšen način pridobiti vsebino dokumenta. S stališča razvijalca programske opreme, takšni artefakti so del notranje podatkovne strukture in ni nujnosti, da bi ponudnih podal na voljo API funkcionalnosti za njihov zajem. Dejansko obstaja zasebni komunikacijski protokol med klientom in strežnikom spletne aplikacije, ki je uporabljen vzporedno ob javnem protokolu.

Preostanek razprave se osredotoča na zajem in analizo podatkov tako imenovanih cloud-native artefaktov v uporabi Google Docs spletne storitve.

3. KAKO DELUJE GOOGLE DOCS?

Za razumevanje pristopov, ki jih lahko uporabimo za forenzično analizo aplikacij v oblaku je potrebno najprej razumeti kako delujejo. Za primer smo izbrali razširjen spletni urejevalnik dokumentov, ki teče v oblaku - Google Docs. Gre za sistem, ki poleg urejanja besedil, preglednic in predstavitev posameznemu uporabniku omogoča tudi interaktivno sodelovanje in hkratno delo na istem dokumentu, kar je izvedeno v realnem času.

Google Docs ima torej številne lastnosti sodobne aplikacije v oblaku, poleg tega pa ima široko bazo uporabnikov, tako za zasebno rabo kot tudi v poslovnih okoljih. Posledično je tudi razumevanje te in podobnih aplikacij nujno za kvalitetno forenzično analizo dokazov, ki ostanejo na voljo pri uporabi takega orodja.

3.1 Prikaz dokumenta in hranjenje zgodovine

Leta 2010 je Google predstavil novo verzijo *Dokumentov*, ki je dala večji pomen sodelovanju v realnem času, s tem namenom pa so tudi spremenili prikaz in izrisovanje elementov izvedli z sistemom Kix. Opustili so urejanje HTML elementov in uporabili klasičen pristop k prikazu vsebine, podoben klasičnim namiznim aplikacijam. Še ena pomembna novost, ki so jo s tem uvedli in ima tudi pomembne implikacije za forenziko je shranjevanje celotne zgodovine urejanj dokumenta, kar poleg prednosti za uporabnika prinaša tudi dodatno vrendost za forenzike, saj lahko pridobijo več informacij kot v primeru če te zgodovine ne bi imeli.

Samo shranjevanje zgodovine ne poteka na način hranjenja posnetkov stanja (ang. Snapshots), temveč se hrani celotna zgodovina akcij od kreacije dokumenta naprej. Ta način omogoča, da nobena izmed akcij, niti brisanje ne more nepovratno izbrisati dela dokumenta, saj se lahko vrnemo v katerokoli točko. Kot že omenjeno ta implikacija prinaša pomembne novosti na področje forenzike, vpliva pa tudi na področje zasebnosti.

Kako natančna je zgodovina priča podatek, da so pri urejanju dokumenta uporabnikove akcije na strežnik poslane tudi do 5 krat na minuto. Akcije se na strežniku ustrezno obdelajo in shranijo, ter pošljejo morebitnim drugim uporabnikom, da se zagotavlja konsistentno stanje za vse udeležence. Testni dokument uporabljen v članku je tako vseboval preko 1000 inkrementalnih sprememb, in šest glavnih revizij. Kdaj pride do nove glavne verzije ni povsem enolično določeno.

Za interno hranjenje dokumenta Google uporablja JSON objekt imenovan dnevnik sprememb (ang. changelog), ki je viden na Sliki 3.1. Struktura je globoko gnezdena, posamezna revizija pa ima svoje polje z pari ključ - vrednost. Vsaka revizija je identificirana z naslednjimi podatki - časovni žig, Google ID avtorja, številka sejne revizije in sama revizija. Nova seja se generira z vsakim odprtjem dokumenta, vanjo pa se potem dodajajo akcije, ki jih izvrši uporabnik. Ključi v slovarju so skrajšani na 2-4 znake, vendar avtorji originalnega članka predvidevajo, da ne z namenom skrivanja (obfuscation), temveč zaradi performančnih razlogov.

Dnevnik sprememb vsebuje poseben objekt, ki hrani segmentiran posnetek stanja v času iz katerega je možna popolna rekonstrukcija začetnega stanja dokumenta. Za vsako revizijo dokumenta objekt vsebuje niz v *plaintext* obliki, ki hrani ves tekst v dokumentu, sledijo pa mu stili in ostali elementi sistema Kix.

3.2 Pridobivanje podatkov iz oblaka

Uporabnikovi podatki se torej ne nahajajo več na njegovem računalniku oziroma trdem disku, ampak se tja prenesejo le na zahtevo in se pri njem načeloma ne hranijo trajno. Eno izmed prvih orodij, ki je bilo prilagojeno novemu tipu oblačnih aplikacij in *SaaS* poslovnemu modelu je bilo orodje *DraftBack*. Gre za brskalniški vtičnik, ki primarno ni namenjen forenzikom, temveč pisateljem, da lažje ocenijo svoje pisanje in ga analizirajo. Potreba po takem orodju pa izhaja tudi iz forenzike, saj nam razumevanje dokumenta in pregleden prikaz zgodovine njegovega urejanja omogočata pridobitev dokazov. *DraftBack* torej ne išče artefaktov in znakov uporabe na uporabnikovem računalniku, ampak se v celoti osredotoča na pridobivanje podatkov iz oblaka. Forenzik tako lahko z ustreznimi uporabnikovimi podatki za dostop neodvisno od lokacije pridobi celotno zgodovino urejanj in vsebino dokumenta.

Prav *DraftBack* je bil osnova za orodje *Kumodocs*, ki je v Pythonu razvito orodje za forenzično analizo Google Docs aplikacije. Ker je bil *Kumodocs* posebej z mislijo na forenziko upošteva vse aspekte, specifične za pridobivanje dokazov, zato gre za bolj uporabno orodje na področju forenzike kot *DraftBack*, ki ni bil razvit specifično za ta namen. Orodje uporablja Google Drive API, preko katerega pridobiva zgoraj omenjene JSON objekte - dnevnike sprememb. Za lažjo nadaljno obdelavo in analizo jih spremeni v CSV format.

Predstavljeni orodij kažeta zasuk, ki se je zgodil ob prehodu aplikacij v oblak - ne le da se zdaj tam nahajajo uporabnikovi podatki, da tam poteka procesiranje in shranjevanje - tja se je preselilo tudi težišče forenzične preiskave, ko govorimo o preiskovanju oblačnih dokumentov. Poleg tega forenzik v veliki meri ni več odvisen od uporabnikove strojne opreme, saj lahko podatke z njegovimi dostopnimi podatki pridobi od kjerkoli ima dostop do interneta, poleg tega pa je pridobivanje natančne zgodovine dokumenta načeloma lažje in bolj uspešno, če ga primerjamo z raziskovanjem zgodovine

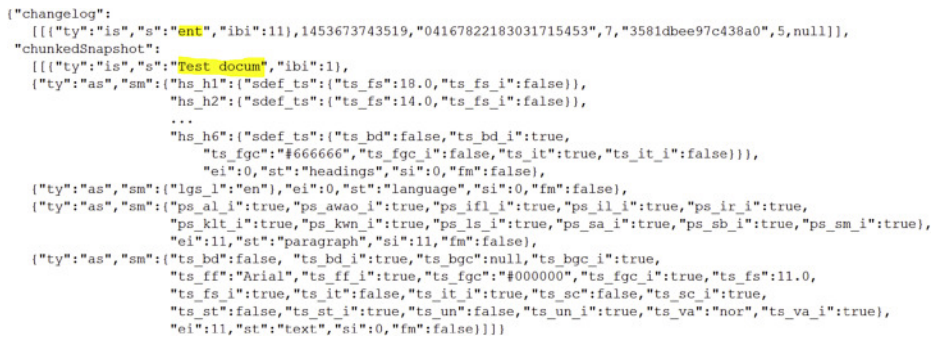


Figure 2: Dnevnik sprememb

klasičnih dokumentov na osebnih računalnikih.

3.3 Prikazovanje slik

Slike in ostali grafični elementi v Google Docs niso neposredno vključeni v JSON objekte (dnevnike sprememb), ampak so v njih le referencirani. Slikovni element tako denimo vsebuje URL lokacije, dostopen preko HTML5 FileSystem APIja. Za shranjevanje slik, ki jih vstavimo dokument Google uporablja CDN (Content Delivery Network) strežnik. Gre za geografsko razporejene strežniške, ki hranijo podatke čim bližje uporabniku, kar omogoča hitrejše nalaganje in bolj učinkovito razpolaganje z viri. Zanimivo je, da če vstavimo fotografijo iz svojega Google Drive diska, bo ta nemudoma prenešena na CDN strežnik in v dokumentu referencirana z povezavo, ki kaže na CDN strežnik in ne na naš disk. S tem tudi ne more priti do problema, da bi uporabnik sliko izbrisal na svojem Driveu, in bi izginil tudi vir za povezavo v dokumentu.

So pa raziskovalci v izvirnem članku prišli do zanimive ugotovitve, ki ima tudi pomembne forenzične implikacije, pa tudi implikacije povezane z zasebnostjo. Z analizo prometa so namreč ugotovili, da dodajanje slike v dokument ustvari povezavo do CDN strežnika. Tudi ob odstranitvi slike iz dokumenta je bila slika še vedno dosegljiva na CDN strežniku tudi brez avtentikacije, potrebno je bilo le poznati povezavo.

3.4 Potek eksperimenta

Raziskovalci so eksperiment, s katerim so potrdili zgoraj opisano arhitekturo sistema za dodajanje grafičnih elementov potrdili na sledeč način.

- V nov dokument so vstavili dve na novo kreirani fotografiji, ki nista bili objavljeni še nikjer na spletu
- Eno izmed fotografij so vključili iz lokalnega diska, drugo pa iz Google Drive storitve
- Z analizo prometa so izluščili povezavi do teh dveh fotografij, ki sta se shranili na CDN strežnik
- Fotografiji so odstranili iz dokumenta, in s skripto 72 ur preverjali, če sta še dosegljivi preko povezave na CDN strežniku
- V drugem delu eksperimenta so prav tako dodali dve fotografiji in nato dokument popolnoma izbrisali.
- Nato je potekalo preverjanje, kdaj bosta fotografiji odstranjeni iz CDN strežnika

3.5 Rezultati in forenzične in zasebne implikacije

Ko sta bili prvi fotografiji dodani, sta se prenesli na CDN strežnik, ta pa je vrnil povezavo, ki so jo vključili v dokument, ki se je vključila v testni dokument. Tudi ob brisanju sta se fotografiji ohranili na strežniku, saj je to potrebno zaradi možnosti obnovitve ob vrnitvi na prejšnje verzije dokumenta. Nekoliko bolj presenetljivo je, da CDN strežnik ne zahteva avtentikacije, ampak se zanaša samo na dolg in naključen URL naslov. Kljub temu, da ta pristop vsebuje določen nivo varnosti, morda za uporabnike ni najbolj intuitiven, kar lahko privede do obnovitve fotografij in drugih elementov, za katere uporabnik predvideva, da so že dolgo nazaj odstranjene.

To je pomembno tudi iz stališča forenzike in zasebnosti. Pričakovana stopnja zasebnosti je namreč višja kot dejanska, če uporabnik ni seznanjen z delovanjem *Dokumentov*, s tem pa lahko forenziki odkrijejo dokaze, za katere je osumljenec prepričan, da so že dolgo tega uničeni.

Fotografije se tako odstranijo šele, ko niso referencirane v nobenem dokumentu več (niti v zgodovini), pa tudi tukaj je drugi del eksperimenta razkril, da akcija brisanja (ang. Garbage collection) ni takojšnja, ampak lahko traja tudi več ur. Če torej osumljenec izbriše dokument z vključenimi fotografijami ali drugimi elementi obstaja možnost, da ob zadosti hitrem posredovanju forenzikov kljub temu odkrijemo določene artefakte.

3.6 Aplikacija Slides

V prejšnjih poglavjih smo se osredotočili na predstavitev urejanja tekstovnih dokumentov, poleg tega pa Google Docs omogoča tudi urejanje predstavitev (*Slides* in preglednic (*Sheets*.

Urejanje predstavitev na nivoju aplikacije poteka na podoben način kot urejanje tekstovnih dokumentov z uporabo dnevnika sprememb. Podatki se torej prenašajo preko JSON objektov, nato pa so uporabniku prikazani z uporabo Javascripta. Del dnevnika sprememb je viden na Sliki 3.6. Glavne spremembe v primerjavi z urejanjem tekstovnih dokumentov je v številčenju in identificiranju elementov, ki se

{"changelog":[[[4,[[1,[365760,205740],[274320,365760]],[45,[],[0,"en"]],[13,0,1,"m","l"],[13,0,2,null,"m"
],[3,"n:slide",158,[2.032025098800659,0.0,0.0,1.1430000066757202,15252.0,27432.0],[55,0,54,16],"n"],[3,"n
:text",108,[1.8287999629974365,0.0,0.0,1.3716000318527222,27432.0,173736.0],[55,1,44,0,54,1],"n"],[41,"n:
text",null,"bodyPlaceholderListEntity"],[42,"n:text",null,"bodyPlaceholderListEntity",0,[13,17,19,21,22,2
3,24,25,26,28,29,31,32,33,36,37],[]],[42,"n:text",null,"bodyPlaceholderListEntity",1,[13,17,19,21,22,23,2
4,25,26,28,29,31,32,33,36,37],[]],[42,"n:text",null,"bodyPlaceholderListEntity",2,[13,17,19,21,22,23,24,2
5,26,28,29,31,32,33,36,37],[]],[42,"n:text",null,"bodyPlaceholderListEntity",3,[13,17,19,21,22,23,24,25,2
6,28,29,31,32,33,36,37],[]],[42,"n:text",null,"bodyPlaceholderListEntity",4,[13,17,19,21,22,23,24,25,26,2

Figure 3: JSON objekt aplikacije Slides

nanaša na samo zgradbo. Tako ima vsaka nova prosojnica svojo številko, naslov in podnaslov, ter iz tekstovnih polj, od katerih ima vsak svoj identifikator.

Dodajanje nove strani v prosojnice sestavlja več manjših združenih operacij. Nekater operacije kot so spreminjanje teme, brisanje ali dupliciranje prosojnic zahtevajo večjo količino prometa.

3.7 Aplikacija Sheets

Pri analizi aplikacije Sheets, je bil potreben drugače pristop kot pri Doucments in Slides. Za razumevanje protokola, po katerem se prenašajo podatki, je bilo potrebno opazovati omrežje. Sheets omogoča revizijo dokumentov za vsako spreminjanje. Kodiranje HTML dokumenta se izvede na strani strežnika, brskalnik pa samo prikaže dokument. Možno je tudi izvoziti stanje preglednic, po vsaki posodobitvi, vendar občuljive informacije – kot so fomule, niso dostopne za spreminjanje. Rešitev za ta probelm je uporaba Google Sheets API, ki ponuja dostop do vsebine posameznih celic. To presega namen te raziskave zaradi drugačne protokola uporabljenega v Sheets, zato ga bomo zanemaili. Rezultat API klicev je pridobiti vsebino celic kodiranih v XML z uporabo Atom Syndication Fromat. Zahteva bolj kompleksno razčlenjevanje (parsing) kot enostaven JSON format, uporabljen v Documents in Slides.

3.8 Predloge in komentarji

Predloge se ustvarijo, kadar je bil dokument spremenjen. Pri Microsoft Word uporabljajo za sledenje spremembam *track changes*. Zapisani so v *changelog*, obravnavajo pa se podobno kot ostale spremembe, vendar omogočajo urejevalniku smiselno oblikovanje uporabniškega vmesnika.

Komentarji niso izrecno zapisani v *changelogu*, venda jih najdemo zapisane pod značko *id*. Google Drive API uporablja seznam metod, ki omogoča pridobitev vseh komentarjev, ki so povezan s tem dokumentom. Ohranjeni so samo aktualni, medtem ko izbrisani nimajo več vsebine. Komentarji in predlogi so del dolgoročne zgodovine dokumenta in se lahko obnovijo bodisi s pomočjo javnega ali zasebnega servisa.

3.9 PoC orodje: kumodocs

Na podlagi rezultatov raziskave je bilo razvito orodje *kumodocs*, ki deluje nad Googlovimi aplikacijami Documents in Slides. Na podlagi *changelog* pridobi in pripravi čistopis dokumenta zadnje verzije ter vse vdelane slike, ki so bile aktivne vsaj nekaj časa.

Za aplikacijo Slides, se vsi tekstovni popravki iz posamezne strani shranijo v svojo datoteko: slide0box0.txt, slide1box1.txt, slide2box2.txt, slide3box3.txt, in tako dalje. Prav tako pridobi vse komentarje, ki so povezani s tem dokumentom.

3.10 Povzetek

Pri analizi treh Google Docs aplikacij, se je začetna hipoteza potrdila. Namreč, ugotovljeno je bilo, da orodja uporabljajo in ohranjajo notranje stanje artefaktov ter jih spreminjajo na ne trivialen način. Ker pa se vse tri aplikacije med seboj razlikujejo, predpostavljano, da so jih razvile tri različne razvojne skupine.

V aplikaciji Documents se podatki prenašajo v podatkovni obliki JSON. Ta format je najbolj splošen in enostaven zato ga uporablja tudi večina spletnih aplikacij. Protokol pri aplikaciji Slides je zasnovan kompleksno in ga je težko razumeti. Omogoča veliko hitrega kodiranja in pretvarjanja podatkov v podatkovno strukturo JSON, kar omogoča odjemalcu hitro iterpretiranje. Tako kot Documents se podatki posredujejo odjemalcu, ta pa poskrbi za usrezen prikaz. Pri aplikaciji Sheets pa se celotno delo opravi na strani strežnika.

Potrebno bi bilo razviti orodja in nove formate podatkov, omogočajo pridobivanje, interpretiranje in dolgoročno shranjevanje artefaktov v oblaku. Orodje bi bilo podobno kot že obstoječ *Draftblack*, ki pa bi bilo sposobno tudi sprotnega zapisovanja teksta z uporabo odprtokodnega urejevalnika *Quill*.

4. RAZPRAVA

Analiza je pokazala, da obstaja v Google Docs več stvari, ki bi bile zanimive za forenzično preiskavo:

Online preview. Google Docs ima vgrajeno revizijo vseh odstopov in sprememb dokumenta. Zgodovina se shranjuje vse od kreiranja dokumenta in ob vsaki spremembi ali dostopu, kar omogoča ponovno vračanje na predhodno verzijo. Preiskovalci pa lahko pregledujejo dokumente v urejevalnem načinu, saj je mogoče zlahka razveljaviti vse nastale spremembe. Vendar obstaja problem, če bi izbrisali celoten dokument. V brskalniku je potrebno nastaviti razširitveno aplikacijo *write blocker*, ki filtrira HTTP zahtevke, ki bi lahko pokvarili dokazno gradivo.

The golden hour. Ideja zlate ure je, da CDN strežniki hranijo dokumente še eno uro po izbrisu. Možno je obnoviti dokumente, ki so bili izbrisani pred nekaj minutami, z kombinacijo metod iz brskalnika in spominske forenzike.

Reverse engineering je še vedno potreben pri forenzični preiskavi v oblaku, vendar se začenja počasi prehod na omrežne protokole. Rousseva raziskava je pokazala, da je javni API dragocen vir dokazov. Potrebno je razumeti delovanje Saas aplikacij, njihove lastne protokole in podatkovne strukture. Na srečno je to *greybox* (in ne blackbox) analiza, pri kateri lahko spremljamo vso komunikacijo in lahko upravljamo z odjemalčevo (JavaScript) kodo na kritičnih trenutkih.

Dolgoročno shranjevajne. Eden izmed večjih problemov pri forenzični analizi v oblaku je dolgoročno shranjevanje podatkov ter jih ohraniti dokler niso ustrezno obdelani. Trenutno je *changelog* nenadomestljiv vir dokazov, vendar pa ga je potrebno ustrezno analizirati. Za razliko od navadnih aplikacij, ne moremo ohraniti kode aplikacije v celoti, saj se razdeli med strežnikom in odjemalcem.

Kako je pri Google Docs? Da bi spremljali nadalnje delo, je pomembno razumeti, kako deluje Google Docs, saj vsebuje celotno zbirko orodij za delo v oblaku. Opravljenih je bilo nekaj predhodnih študij na orodjih: Zoho Writer, Microsoft Word Online in Dropbox Paper. Opazka je, da imajo kljub sprotnim posodobitvam uporabniki celoten dostop do vseh verzij dokumentov. Samo pri Google Docs je bi opažen API, s katerim je mogoče pridobiti vse dostope do določenega dokumenta. Ostali imajo na strežniku sezname, ki dokazujejo kaj je uporabnik sproti spreminjal in katero verzijo uporablja. Na primer pri Zoho Writer, se pri obnavljanju na starejšo verzijo doda zapis v seznam *user-selectable version*. Indikator verzije pa kaže na verzijo, ki jo je uporabnik izbral. Word in Paper imata podoben koncept.

5. ZAKLJUčEK

Da bi razumeli izzive in priložnosti, ki jih artefakti v oblaku ponujajo, je bil narejen pregled Google Docs. Definicija takšnih artefaktov kot podatkovnih objektov, ki jih uporabljajo aplikacije *SaaS* niso shranjeni na odjemalčevi napravi.

Formiranje problema. Tradicionalni pristopi na strani odjemalca so popolnoma slepi za pridobivanje podatkov iz artefaktov v oblaku. Notranja podatkovna struktura v artefaktu vsebuje pomembne zgodovinske podatke. Potrebno je razviti forenzično orodje, s katerim bi se lahko ti podatki zajeli in ustrezno obdelali ter orodje za prikaz zgodovine artefaktov in njihovo arhiviranje.

Artefakti in analiza. Opravljena je bila analiza Google Docs, na aplikacijah Documents in Slides ter njihovih *changelog* v notranjih strukturah. Analizan je sledila Somerjevi začetni analizi in sistematičnemu dokumentiranju najdenih rezultatov. [6] Poleg tega se je iskali tudi mehanizmi za vstavljanje predmetov v artefakte. Izkazalo pa se je, da je Google-ov CDN strežnik ogromen vir informacij, z navidez neomejenim časovnim obdobjem.

Razvojno orodje PoC. Praktični rezultat te raziskave je razvoj orodja za zajem in obdelavo zgodovine dokumentov Documents in Slides. Možnost izvozi vsebina besedila, za vsako verzijo, vgrajene slike, risbe ter zgodovino pripomb povezanih z dokumentom. Orodje se imenuje *kumodocs* in je na voljo na GitHub: https://github.com/kumofx/ kumodocs.

Orodje ni namenjeno samo forenzični preiskavi, lahko se ga uporablja tudi na preiskavah na področju zasebnosti za revizijo dokumentov. V bližnji prihodnosti se pričakuje zaključene analize ostalih aplikacij iz zbirke Google Docs, ter razvoj orodja, ki bi imel možnost pregledovanja, pridobivanja in dolgoročnega ohranjevanja dokazov iz Google Docs.

6. ZAHVALA

Zahvaljujemo se prof. dr. Andreju Brodniku in as. dr. Gašperju Fele Žoržu za pomoč pri pripravi članka.

7. VIRI

[1] M. B. Q. Chen J. Quill rich text editor. `https://github.com/quilljs/quill/`.
[2] H. Chung, J. Park, S. Lee, and C. Kang. Digital forensic investigation of cloud storage services. *Digital Investigation*, 9(2):81 – 95, 2012.
[3] Google. Google drive blog archive.
[4] Google. The next generation of google docs.
[5] J. S. Hale. Amazon cloud drive forensic analysis. *Digital Investigation*, 10(3):259 – 265, 2013.
[6] S. J. How i reverse engineered google docs to play back any documents keystrokes.
[7] B. Martini and K.-K. R. Choo. Cloud storage forensics: owncloud as a case study. *Digital Investigation*, 10(4):287 – 299, 2013.
[8] D. Quick and K.-K. R. Choo. Dropbox analysis: Data remnants on user machines. *Digital Investigation*, 10(1):3 – 18, 2013.
[9] D. Quick and K.-K. R. Choo. Google drive: Forensic analysis of data remnants. *Journal of Network and Computer Applications*, 40:179 – 193, 2014.
[10] V. Roussev and S. McCulley. Forensic analysis of cloud-native artifacts. *Digit. Investig.*, 16(S):S104–S113, Mar. 2016.

Forenzična raziskava primerov spletnega zalezovanja s pomočjo analize vedenjskih razvidov

Biljana Miceva
Fakulteta za računalništvo in informatiko
Ljubljana, Slovenija
bm1795@student.uni-lj.si

Karmen Knavs
Fakulteta za računalništvo in informatiko
Ljubljana, Slovenija
kk7072@student.uni-lj.si

Nika Eržen
Fakulteta za računalništvo in informatiko
Ljubljana, Slovenija
ne8927@student.uni-lj.si

POVZETEK

Z razvojem spleta in vseprisotnostjo računalništva je spletno zalezovanje postalo resen zločin. Metode analize vedenjskih razvidov pomagajo pri razumevanju zločinca, žrtve, prizorišča zločina in dinamike zločina. Spadajo pod deduktivne metode. Statistični podatki kažejo, da so najpogostejše žrtve mlade ženske, motiv zalezovalca pa je največkrat ljubezenski. Predstavljena je forenzična analiza 20 primerov, ki jih je preiskovala Dubajska policija. Proces analize digitalnih podatkov je bil iterativen ter progresiven. Pri primerjavi s statističnimi podatki na večjih množicah smo našli razlike. Rezultati so pokazali, da metode analize vedenjskih razvidov omogočajo boljše razumevanje in interpretacijo obnašanja žrtve in zločinca. Veliko držav še sprejema dodatne zakone, ki pokrivajo spletno zalezovanje.

Ključne besede

Spletno zalezovanje, analize vedenjskih razvidov, profiliranje.

1. UVOD

Spletno zalezovanje je zalezovanje oziroma nadlegovanje preko različnih telekomunikacijskih sredstev in je relativno nova oblika zločina. V zadnjih letih je bilo zabeleženo povečanje primerov spletnega zalezovanja, tako so npr. v Dubaju zaznali 39% porast takšnih primerov v letih 2010-2014 [17]. Številne študije o spletnem zalezovanju kažejo, da je to široko razširjena vrsta spletnega nadlegovanja, ki je slabo razumljena in obravnavana [2, 4].

V tem članku se bomo osredotočili na reševanje primerov spletnega zalezovanja s pomočjo uporabe metod analize vedenjskih razvidov (AVR). Metode AVR spadajo pod deduktivne metode profiliranja. Namen uporabe metod AVR je iz dokaznega gradiva izpeljati lastnosti in karakteristike osumljenca in tako hitreje in učinkoviteje zaključiti primer. Trenutno obstaja nekaj literature na temo uporabe metod AVR pri reševanju spletnih zločinov [19, 12, 18], le malo pa je primerov uporabe. Metode AVR so tako preizkusili kot orodje za prepoznavo karakteristik uporabnikov družabnega omrežja Facebook [28]. V članku Forensic investigation of cyberstalking cases using Behavioural Evidence Analysis [17], ki je tudi podlaga za ta članek, so avtorji metode AVR preizkusili na pravih primerih spletnih zločinov.

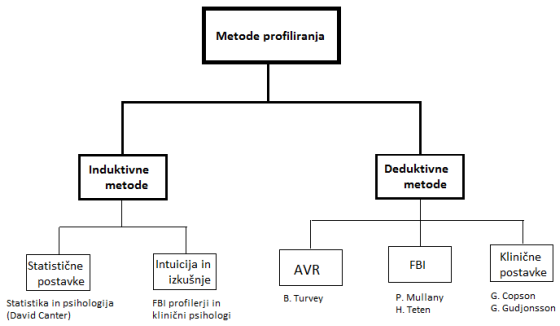
V razdelku *AVR in spletno zalezovanje* sta na širše razložena v naslovu omenjena pojma. V razdelku *Statistika* je s številkami predstavljena trenutna situacija spletnega zalezovanja (koga zasleduje kdo, zakaj, koliko časa ...). V razdelku *Metodologija* pa si bomo ogledali, kako so v Dubaju s kombinacijo tehnik digitalne forenzike in metod AVR rešili primere iz realnega življenja [17].

2. AVR IN SPLETNO ZALEZOVANJE

V tem razdelku sta opisana pojma AVR in spletno zalezovanje, ki ju uporabljamo skozi celotni članek.

2.1 Analize vedenjskih razvidov

Analize vedenjskih razvidov (AVR) so ene izmed deduktivnih metod profiliranja. Profiliranje se v splošnem deli na induktivne in deduktivne metode. Z induktivnimi metodami preko statističnih informacij posplošimo primer. Z deduktivnimi metodami pa značilnosti storilca razberemo iz vsakega primera posebej. Prednost uporabe deduktivnih metod je, da se ne zanašamo na velikokrat netočne ali celo lažne statistične informacije, in se raje posvetimo konkretnemu primeru in njegovim podrobnostim. Hkrati je to tudi največja ovira pri uporabi deduktivnih metod, saj se mnogokrat soočamo z nezadostnimi količinami informacij o primeru [21].



Slika 1: Razdelitev metod profiliranja.

Z metodami AVR skušamo preko forenzičnih dokazov in samega kaznivega dejanja določiti lastnosti storilca kaznivega dejanja. Na ta način lahko prepoznamo možne spremembe v storilčevem razmišljanju in obnašanju.

Spletno zalezovanje je relativno nova oblika zločina in zatorej ne obstaja veliko (zadostnih) statističnih informacij. Ker veliko žrtev spletnega napada ne zazna, bi tudi ti primeri izostali iz statistike in jo na ta način popačili. V primerih spletnega zalezovanja je tako bolj primerno uporabiti metode AVR, saj se le te ne zanašajo na statistiko, ampak obravnavajo vsak primer posebej, striktno glede na zbrano dokazno gradivo.

Metode AVR sestojijo iz štirih korakov [7, 25]:

- **Dvoumna forenzična analiza** je proces objektivnega ocenjevanja razpoložljivih dokazov, neodvisno od interpretacije drugih, katerega namen je določiti pravi pomen dokazov. Cilj je zaznati, če je bilo med do tedaj izvedeno preiskavo kaj spregledano, oziroma, če so bile storjene kakšne napake.

  Primer: Če osumljenčev računalnik uporablja več oseb, lahko to močno vpliva na potek preiskave. Če se med preiskavo ne ugotovi, da računalnik uporablja več oseb, to drastično zniža verjetnost, da so ugotovitve preiskave pravilne, saj bodo kasneje v preiskavi karakteristike večih oseb upoštevane kot karakteristike ene same osebe.

- **Viktimologija** preučuje značilnosti žrtve (npr. fizične značilnosti, zakonski status, stil življenja). Cilj je ugotoviti, zakaj je bila napadena prav ta oseba. Če je vzrok napada najden, lahko na tem koraku ugotovimo tudi morebitne povezave med žrtvijo in storilcem.

  Primer: Računalniški vsiljivec je morda poskušal dostopati do računalnika. Zapisi o neuspelih poskusih lahko implicirajo, da storilec ni dobro seznanjen z napadenim računalnikom. Pomanjkanje teh zapisov pa lahko kaže na to, da je storilec dobro seznanjen s tem računalnikom ali, da zna svoje sledi dobro skriti.

- **Karakteristike kraja zločina.** Med sistematičnim pregledom kraja zločina se pokažejo določeni vidiki in vzorci storilčevega obnašanja. Posebej ločimo obnašanje, ki je potrebno za izvedbo zločina (modus operandi ali usmerjeno obnašanje) in obnašanje, ki ni potrebno za izvedbo zločina (motivacijsko usmerjeno obnašanje). Na podlagi karakteristik zločina lahko določimo npr. število oseb, ki so zločin izvedle.

  Primer: Če preiskovalci odkrijejo, da je vsiljivec vdrl v več računalnikov na omrežju, lahko potencialno poiščejo več dokazov. Informacije, ki jih je vsiljivec iskal pa lahko odkrijejo njegov motiv.

- **Karakteristike storilca.** Na tem koraku preiskovalec združi rezultate prejšnjih treh korakov in tako določi najbolj verjetne vedenjske in osebnostne karakteristike storilca. Na podlagi teh domnev pa sestavi ustrezen profil storilca.

  Primer: Recimo, da so preiskovalci s pomočjo beležk (log datotek) ugotovili, da je storilec naredil kopijo digitalne datoteke. Če storilec beležk ni izbrisal, je na preiskovalcih, da si postavijo naslednja vprašanja. Zakaj je storilec pustil tako pomemben dokaz? Ali se storilec ni zavedal, da se podatki hranijo v beležkah? Ali storilec ni uspel izbrisati datotek? Odgovori na tovrstna vprašanja lahko pomagajo odkriti določene morebiti koristne karakteristike storilca.

2.2 Spletno zalezovanje

Spletno zalezovanje (angl. cyberstalking) je zalezovanje, oziroma nadlegovanje preko različnih telekomunikacijskih sredstev. Bolj natančno ga v literaturi opredelijo kot obnašanja, kjer ena ali več oseb uporablja informacijsko tehnologijo (npr. spletna pošta, socialna omrežja) za ponavljajoče se nadlegovanje druge osebe ali skupine, z namenom, da bi le tej vzbudila strah, nelagodje in občutek ogroženosti [11, 5]. Pod tovrstna obnašanja sodijo grožnje, lažne obtožbe, nadzor in lažno predstavljanje.

V primerjavi z običajnim zalezovanjem, kjer storilec žrtev neprestano opazuje, zasleduje in nadleguje, je spletnemu zalezovanju dodana nova dimenzija. Z uporabo informacijske tehnologije lahko storilec žrtev konstantno nadleguje tudi na dolge razdalje, brez da bi izdal svojo identiteto [11]. Ker je storilec oddaljen od žrtve tako fizično kot čustveno (ne vidi njenega odziva), se storilec lahko obnaša popolnoma drugače, kot bi se obnašal, če bi žrtev nadlegoval neposredno [14, 3]. Storilcu moderna tehnologija omogoča tudi ustvarjanje večih spletnih osebnosti, kar lahko dodatno oteži preiskavo primera [24].

McFarlane in Bocij [16] sta na podlagi 24 intervjujev žrtev spletnega zalezovanja spletne zalezovalce razdelila v štiri kategorije:

- **Maščevalni spletni zalezovalci** neprestano nadlegujejo svojo žrtev brez posebnega razloga. Pogosto trpijo za psihološkimi motnjami.

  Primer: Nepomemben prepir na spletu (npr. komentarji pod sliko objavljeno na družabnem omrežju Facebook) se razvije v spletno zalezovanje.

- **Zbrani spletni zalezovalci** zalezujejo brez želje da bi vzpostavili stik z žrtvijo, želijo ji le vzbuditi občutek konstantnega nelagodja.

  Primer: Luke J. Heller je objavil več slik svoje tedajšnje žene na sadomazohistične spletne strani skupaj z njenimi kontaktnimi informacijami in neprimernim povabilom. To je storil, da bi se žena počutila osramočeno, on pa je na ta način sprostil svoje frustracije [10].

- **Intimni spletni zalezovalci** želijo pridobiti čim več pozornosti žrtve. Običajno imajo podrobno znanje o osebi, ki jo zalezujejo.

  Primer: David Heiss je več let zalezoval Joanno Witton preko spleta, ker se je zaljubil vanjo preko foruma. Njegova obsedenost je privedla tako daleč, da je umoril Joanninega fanta [27].

- **Kolektivni spletni zalezovalci** so skupina posameznikov, ki žrtve nadlegujejo z uporabo komunikacijske tehnologije.

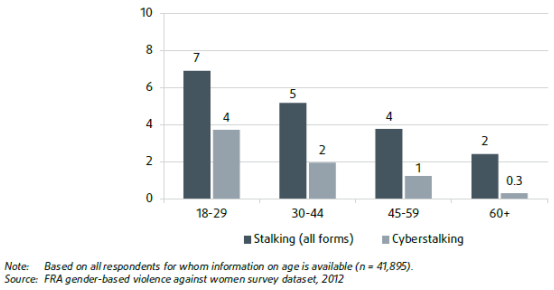
Poudariti je treba, da se zgornja kategorizacija ne uporablja pri metodah AVR, saj so kategorije določene samo iz zornega kota žrtve ne pa tudi storilca in same narave primera.

3. STATISTIKA

Spletno zalezovanje je hitro napredujoča oblika nadlegovanja in na žalost mu veliko zakonodaj ne sledi v praksi [9]. Uradni statistični podatki v povezavi z njim so skopi in neažurni. Veliko člankov se zato opira na svoje raziskave, najpogosteje spletne ankete. Nekatere se opirajo na podatke zalezovalcev [15] ali pa izvajajo raziskavo v zavarovanih prostorih (na Dubajski policiji [17]). Enotne definicije spletnega zalezovanja ni.

V ZDA skoraj 1,5 milijona ljudi trpi zaradi spletnega zalezovanja vsaj enkrat letno. Ena od 12 Američank bo trpela zaradi vsaj enega dogodka, povezanega s spletnim zalezovanjem. Podporna skupina WHOA (Working to Halt Online Abuse) prejme do 75 pritožb povezanih z njim na teden. Povprečno trajanje spletnega zalezovanja bo trajalo dve leti, če pa so odnosi med storilcem in žrtvijo intimni, se trajanje podvoji. Zakonodaje štejejo mednje tudi krajo spletne identitete [9].

Ena vidnejših uradnih raziskav na področju Evrope je bila objavljena leta 2014 s strani Agencije Evropske unije za temeljne pravice (FRA - European Union Agency for Fundamental Rights). Obsegala je 28 članic Evropske unije in se osredotočala na nasilje proti ženskam. V njej je sodelovalo 42000 žensk, ki so odgovarjale na vprašanja povezana s psihičnim, spolnim in fizičnim nasiljem, zajemala pa so tudi spletno zalezovanje [8]. Med ženskami s starostjo od 18 do 29 let je spletno zalezovanje v 12 mesecih pred anketiranjem doživelo 4% sodelujočih, med ženskami starejših od 60 let pa le 0,3% (Slika 2).

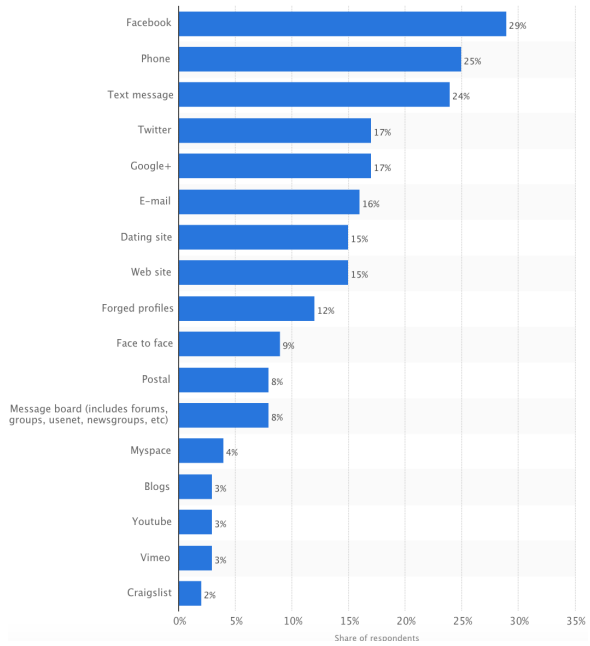


Slika 2: 12-mesečna prisotnost zalezovanja (vseh oblik) in spletnega zalezovanja, glede na starost žrtev.

Uradnih podatkov osredotočenih na Slovenijo nismo našli. V magistrski nalogi objavljeni leta 2015 smo našli rezultate ankete, ki se osredotoča na otroke med 9. in 14. letom - ti zaradi svoje neizkušenosti spadajo med najbolj ranljive skupine [13]. Polovica (51% - 54% otrok od 105) je potrdilo, da jih starši pri uporabi spleta ne nadzorujejo. Zaskrbljujoč je podatek, da se samo 59% otrok ne bi srečalo v živo z neznano osebo. V magistrski nalogi smo opozorjeni, da naš zakonik trenutno še ne opredeljuje pojma spletnega zalezovanja in nadlegovanja [13].

3.1 Naraščanje primerov s spletnim zalezovanjem in njegove oblike

Eden od glavnih vzrokov porasta spletnega kriminala je ta, da ima danes že skoraj vsak svoj računalnik ali pametni mobilni telefon, zato se lahko na svetovni splet povežemo praktično kjerkoli in kadarkoli. Na spodnji sliki (Slika 3) vidimo najpogostejše načine stopnjevanja primerov spletnega zalezovanja, ki jih je WHOA zbrala s pomočjo 256 anketirancev leta 2013. Čeprav je e-pošta najpogostejša oblika spletnega zalezovanja, največ ljudi zmoti zalezovanje na socialnih omrežjih (najpogosteje preko Facebooka) ter preko telefonskih klicev in sporočil [17, 26].



Slika 3: Načini stopnjevanja primerov spletnega zalezovanja leta 2013 [23].

Razlog za naraščanje je tudi odprtost internetnega sveta. 63% Facebook profilov je javno vidnih, 55% najstnikov deli osebne podatke brez kakršnihkoli nastavitev zasebnosti. Okoli 83% študentov redno pregleduje profil svojega bivšega partnerja na družabnem omrežju [9].

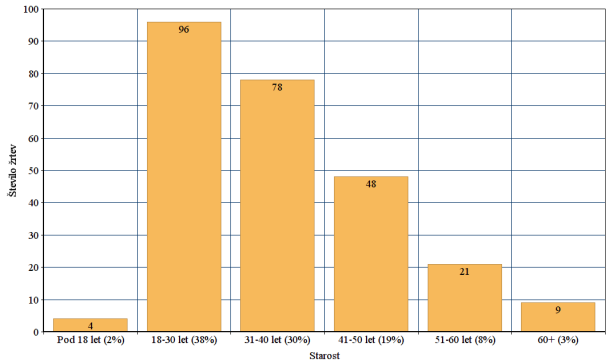
3.2 Karakteristike

Večina zalezovalcev (več kot 50%) odneha prej kot v pol leta. Podatki dokazujejo, da gre pogosteje za zalezovalce, ki so samski in žrtve ne poznajo osebno [15]. Približno 20% vseh vztraja pri zalezovanju več kot eno leto [17, 15].

Pri opisu karakteristik se nanašamo na 256 udeležencev, ki so leta 2013 odgovarjali na anketo organizacije WHOA. Statistika med večjimi vzorci je podobna. V razdelku *Metodologija* primerjamo rezulate iz članka [17], kjer so bili podatki zbrani na policiji v Dubaju.

3.2.1 Starost

Največ žrtev spada v starostno skupino 18 do 40 let (Slika 4).



Slika 4: Delež žrtev glede na starost.

3.2.2 *Spol*

Več žrtev spletnega zalezovanja je ženskega kot moškega spola (Tabela 1).

Tabela 1: Žrtve spletnega zalezovanja glede na spol

Ženske	174	68%
Moški	77	32%

Več zalezovalcev je moškega kot ženskega spola (Tabela 2).

Tabela 2: Spletni zalezovalci glede na spol

Moški	106	40%
Ženske	75	30%
Neznano	75	30%

3.2.3 *Poklic in izobrazba*

Nismo našli izrazitih karakteristik. Prevladuje srednji sloj, vendar to izraža tudi splošna sestava prebivalstva. Za zalezovalce nismo našli podatkov.

3.2.4 *Stan*

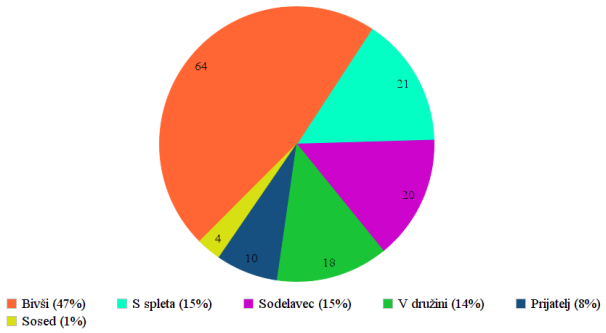
Žrtve so najbolj pogosto samskega stanu (Tabela 3).

Tabela 3: Stan žrtve

Samski	134	52%
Poročen	80	31%
Ločen	23	10%
Ostalo (ovdovel ...)	19	7%

3.2.5 *Razmerje med zalezovalcem in žrtvijo*

Največkrat se pojavlja kombinacija moškega zalezovalca in ženske žrtve. V večini primerov se poznata od prej. Pri podatkih organizacije WHOA to ni izrazito (samo v 53% primerih se poznata od prej). Vrsta prejšnjega razmerja z zalezovalcem je največkrat ljubezensko (Slika 5).



Slika 5: Prejšnje razmerje med zalezovalcem in žrtvijo.

3.3 Napovedovanje obnašanja iz statistike

Obstajajo raziskave, ki s pomočjo statističnih podatkov glede na izpolnitev določenih pogojev (starost, poznavanje žrtve, motivacija) napovedujejo dolžino zalezovanja, oziroma ocenjujejo tveganje za nadaljevanje. Ocenjevanje tveganja zalezovanja mora biti ponavljano na določeno frekvenco [15]. Zelo vztrajni zalezovalci so pogosto stari preko 30, poznajo žrtev in imajo maščevalni motiv ali iščejo intimnost. Velikokrat trpijo zaradi paranoje, zlasti shizofrenije. Pri zalezovalkah je 2,5-krat več možnosti, da imajo psihično bolezen, kot pri moških zalezovalcih.

Potrebno je poudariti, da pri metodah AVR ne uporabljamo statistike. Statistika je v članku navedena le zato, da bralec pridobi vpogled v razširjenost in karakteristike spletnega zalezovanja.

4. METODOLOGIJA

AVR je v teoriji zelo koristna metoda, ki preiskovalcem omogoča razumeti tako profile vseh vpletenih v kaznivo dejanje kot tudi potek le-tega. Ponuja tehnike, ki pomagajo pri pridobivanju informacij iz digitalnih dokazov in pri rekonstrukciji celotnega dejanja. Trenutno ne obstaja dovolj empiričnih raziskav, ki uporabljajo AVR metodo, zlasti na primerih spletnega zalezovanja. Zaradi uporabe vse bolj sofisticirane tehnologije za izvajanje spletnega nadlegovanja, je zelo težko slediti storilcem in iz dneva v dan, sledenje predstavlja zelo velik izziv tudi za najbolj izkušene preiskovalce. Kot posledica tega, ni zadostne strokovne literature, ki se ukvarja s to problematiko.

Prednost pri raziskovanju primerov spletnega zalezovanja je ravno tehnologija oziroma podatki, ki se lahko pridobijo iz storilčeve naprave, in se nato analizirajo z uporabo metod AVR. Na ta način lahko preiskovalci analizirajo profile osumljencev, ustvarijo relacije med vpletenimi, prepoznajo motivacije za dejanja ter ocenjujejo tveganje stopnjevanja spletnega zalezovanja v fizično zalezovanje.

Prednost AVR metod je raziskovalna strategija, zgrajena na podlagi primerov, ki analizirajo dokaze z namenom, da bi izpeljali določene vedenjske ali osebne značilnosti zalezovalca [25].

Metode AVR so se do sedaj že izkazale kot uporabne pri raziskovanju tradicionalnih kriminalnih primerov. Zaradi svoje zasnove pa so koristne tudi pri primerih spletnega zalezovanja. Različni digitalni artefakti, kot so besedilna komunikacija, e-pošta, zgodovina brskanja, izbrisane datoteke, časovni žigi itd., lahko razkrijejo uporabne informacije o vedenju ali značilnostih zalezovalca ali žrtve, kar seveda, lahko na različne načine pomaga preiskovalcu. Posledično se lahko naredi bolje zasnovana rekonstrukcija primera, ki nudi podlago za strokovno razlago. Kot smo že omenili, ne obstaja veliko raziskav ali praks preiskovanja digitalnega kriminala, ki uporabljajo metode AVR, še posebej ne za primere digitalnega zalezovanja. Še manj literature pa obstaja pri uporabi teh metod v pomoč interpretiranja digitalnih dokazov. Lahko izpostavimo primer, kjer sta raziskovalki Silde in Angelopoulou poskusili razviti profil zalezovalca z uporabo metod AVR pri digitalni forenzični preiskavi [22]. Pri tem sta uporabili simulacijo spletnega zalezovanja, ki je bila analizirana z uporabo drugih tehnik poleg metod AVR. Zaključek te raziskave je, da so metode AVR instrument triaže, oziroma, da se AVR metode osredotočajo na lokacije, ki najverjetneje vsebujejo relevantne dokaze. Drugi raziskovalec je predlagal model, ki je vključil metode AVR v proces digitalne forenzične raziskave [20]. Predlagan model vključuje 6 faz, in sicer: klasifikacija primera, statistična analiza, analiza časovnice, vizualizacija, odločitev in mnenje. Vključil je tri študije, da bi izpostavil značilnost časovne analize v forenzičnih raziskavah.

4.1 Izvedba metodologije

V tem delu opisujemo metodologijo uporabljeno za izvedbo študije. Podobna metodologija je bila uporabljena na primerih preiskovanja slik otrok z eksplicitno vsebino. Študija je bila izvedena v laboratoriju Dubajske policije, na oddelku za digitalne dokaze [17]. Avtorji so uporabili forenzično pravilne kopije vsebin diskovja vseh vpletenih.

4.1.1 Izbira primerov

Izbira primerov je bila narejena na podlagi sledečega kriterija: primer je bil vključen v izbor, če je obnašanje storilca kaznivega dejanja, po izkušnjah žrtve, ustrezalo definiciji spletnega zalezovanja, če je storilec uporabljal računalnik kot platformo nezakonitega dejanja in če je bila zagotovljena dostopnost slikovnih datotek.

Vzorec je zajemal 20 primerov različnih variacij spletnega zalezovanja, ki so bili storjeni v Dubaju v letih 2010-2014. Vključenih je bilo 31 računalnikov. Najprej so bili trdi diski vseh računalnikov zaseženi, verificirani in ustrezno arhivirani s strani Oddelka za digitalne dokaze.

4.1.2 Viri podatkov

Primarni podatki za to študijo so bili digitalni podatki, hranjeni v slikovnih datotekah vseh zaseženih računalnikov ter vsi drugi relevantni dokumenti. Najpogosteje v takšnih primerih digitalni forenzik pridobi zasežen disk žrtve in morebitnih ostalih osumljencev. Na ta način raziskava teh naprav poskuša rekonstruirati komunikacijo med zalezovalcem in žrtvijo, prepoznati motivacijo zalezovalca, razumeti okolje, v katerem se je zalezovanje zgodilo in ugotoviti odnos zalezovalec-žrtev.

4.1.3 Zbiranje podatkov in analiza

Pomembno je izpostaviti, da je študija uporabila kombinacijo tehnik digitalne forenzike in metod AVR. Ker je bila uporabljena metoda (AVR) deduktivna, je bilo ključno razumeti kontekst vsakega primera posebej, preden so se analizirali digitalni dokazi. Zato je bila za vsak primer posebej pregledana celotna dokumentacija. Aktivnosti vpletenih so bile analizirane z rekonstrukcijo komunikacije, obnovitvijo artefaktov komunikacij na socialnih omrežjih, obnovitvijo izbrisane e-pošte ter pridobivanjem vseh drugih relevantnih podatkov kot so dokumenti, slike, videi, zapisi registrov, zgodovina brskanja, metapodatki itd. Proces analize pridobljenih podatkov je bil iterativen ter progresiven. Oba uporabljena pristopa sta se pri tem dopolnjevala. Za vsak primer so se na vsakem koraku pridobljeni digitalni dokazi uporabili kot vhodni podatki za metode AVR. Iz izhodnih podatkov metod AVR so nato razbrali možne nove lokacije potencialnih dokazov, pridobili pa so tudi vpogled v karakteristike razmerja žrtev-storilec. Ugotovitve analize so bile povzete za vsak primer posebej. Združene pa so bile podobnosti v podatkovnih virih, interpretacije in značilnosti analize. Sistematično so bili analizirani tudi kvalitativni podatki zbrani tekom forenzične preiskave digitalnih naprav (tako kot poudarjajo drugi avtorji [6]). Tekom analize so preiskovalci pridobili in združili podatke, ki se lahko primerjajo s prejšnjim delom na tem področju. Sem spadajo značilnosti zalezovalca in žrtve, kot so: starost, spol, narodnost, zaposlitveni status, zakonski status, kvalifikacije in računalniška pismenost ter značilnosti specifične za zalezovalca, kot je zgodovina nezakonitega dejanja, kazenska evidenca in zgodovina psihološkega zdravja. Preiskovalci so raziskali tudi kazalce, povezane s takšnim obnašanjem: čas trajanja spletnega zalezovanja, sredstva komunikacije, spletne strani socialnih omrežij, forume itd. Množica kazalcev, povezanih s slogom in vsebino komunikacije: lažno predstavljanje, uporaba slik, izjavljanje ljubezni, komentarji na podlagi seksualne orientacije, grožnje in vsiljivi komentarji ter obrekovanje.

4.2 Rezultati

V tem podrazdelku bomo opisali značilnosti vpletenih, odnos žrtev-zalezovalec ter obnašanje zalezovalca.

4.2.1 Značilnosti žrtve ter zalezovalca

Spodnja tabela (Tabela 4) opisuje značilnosti žrtve ter zalezovalca. Žrtve so bile stare med 23 in 48 let, zalezovalci pa od 21 do 63 let. Skladno s statistiko (tretji razdelek) je največ žrtev mladostnikov.

Večina žrtev je bila ženskega spola (75%), kar se sklada tudi z drugimi raziskavami. Kar zadeva narodnosti, je najmanjši delež pripadal Azijski skupini (Vzhodni in Južni Azijci), iz skupine Bližnjega Vzhoda je bilo kar 45%, iz skupine Kavkazijcev pa 30 %. Pomembno je izpostaviti tudi, da je bila večina zalezovalcev iz te raziskave zaposlenih v času, ko so žrtev zalezovali (80%). Najmanjši odstotek (10%) žrtev je bilo študetov, večina (60%) pa jih je spadala v srednji poklicni razred. Tudi večina zalezovalcev (70%) je bilo iz srednjega poklicnega razreda.

4.2.2 Odnos žrtve in zalezovalca

Spodnja tabela (Tabela 5) opisuje spol in razmerje, ki ga je zalezovalec imel z žrtvijo. Za razliko od statistike in drugih

Tabela 4: Karakteristike žrtev in storilcev spletnega zalezovanja

karakteristike	žrtev	storilec
starost	23-48	21-63
21-30	8/20 (40%)	4/20 (20%)
31-40	6/20 (30%)	7/20 (35%)
41+	6/20 (30%)	9/20 (45%)
spol		
ženski	15/20 (75%)	4/20 (20%)
moški	5/20 (25%)	16/20 (80%)
narodnost		
kavkazijska	6/20 (30%)	7/20 (35%)
srednje vzhodna	9/20 (45%)	8/20 (40%)
vzhodno/južno azijska	5/20 (25%)	5/20 (25%)
poklicni status		
visok	2/20 (10%)	0/20 (0%)
srednje	12/20 (60%)	14/20 (70%)
nizek	3/20 (15%)	4/20 (20%)
študent	1/20 (5%)	0/20 (0%)
nezaposlen	2/20 (10%)	2/20 (10%)

študij [16], je vsaka žrtev imela določeno razmerje z zalezovalcem. 35% so imeli predhodno ljubezensko razmerje, 40% pa so bili sodelavci. Tudi to se razlikuje od statistike, saj prevladuje ljubezensko razmerje. Večina zalezovalcev je bilo moških, kar 80%. Od tega so bile v večini primerov ženske zalezovane s strani moških (60%), kar se sklada s statistiko. V 20% primerov so bili moški zalezovani s strani moških.

Tabela 5: Spol in razmerje storilec/žrtev

razmerje	odstotek
ljubezensko razmerje	7/20 (35%)
znanca	2/20 (10%)
sodelavca	8/20 (40%)
spoznala preko spleta	2/20 (10%)
neznano	1/20 (5%)
storilec-žrtev	
moški-ženska	12/20 (60%)
ženska-moški	1/20 (5%)
ženska-ženska	3/20 (15%)
moški-moški	4/20 (20%)

4.2.3 Obnašanje zalezovalca

Analizirani podatki so pokazali, da je najbolj razširjen način zalezovanja kar preko elektronske pošte, kot je pokazano v tabeli (Tabela 6) v nadaljevanju.

V več kot polovici analiziranih primerov (55%) so zalezovalci uporabljali elektronsko pošto na samem začetku komunikacije z žrtvijo, kar se sklada s statistiko. V 15% so zalezovalci komunicirali z žrtvijo kar preko njihovih osebnih elektronskih naslovov. V 25% primerov se je komunikacija odvijala preko žrtvine poslovne elektronske pošte, v 15% primerov pa so zalezovalci imeli dostop do žrtvinih elektronskih naslovov, kar so izkoriščali za oponašanje žrtev. Drugi najbolj pogost način zalezovanja je bil preko socialnih omrežjih, najpogosteje Facebook ali Twitter. Zalezovalci so izkoriščali socialna omrežja tako, da so objavljali neprimerno vsebino na žrtvinem profilu, ji grozili ali puščali sovražne komentarje.

Tabela 6: Način in dolžina spletnega zalezovanja

način	žrtve
dolžina zalezovanja	
6 mesecev ali manj	12/20 (60%)
7 mesecev-1 leto	4/20 (20%)
2 leti	1/20 (5%)
Neznano	3/20 (15%)
načini zalezovanja	
e-pošta	11/20 (55%)
socialna omrežja	5/20 (25%)
forumi in oglasne table	1/20 (5%)
portali za zmenke	3/20 (15%)

Spletne strani za zmenke so bile največkrat uporabljene za oponašanje žrtve, postavljanje lažnih komentarjev o žrtvinih seksualnih fantazijah in željah ter tako vzpodbujali obiskovalce spletne strani, da žrtev kontaktirajo. V večini analiziranih primerov (60%) je zalezovanje trajalo tri tedne do šest mesecev. Tematska analiza komunikacije med žrtvijo in zalezovalcem je pokazala, da je večina zalezovalcev uporabljala grožnje in nasilne besede ali obsesivne in ljubezenske besede. V 33% primerov je zalezovalec žrtvi ukradel identiteto, ko je žrtev oponašal na spletu. Nobeden od zalezovalcev ni sodeloval ali bodril drugih oseb, naj se mu pridružijo v zalezovanju. Večina zalezovalcev, ki je pokazala maščevalno obnašanje, je bila ali prejšnji zakonski partner žrtve ali nezadovoljen zaposlen. V nadaljevanju so v tabeli (Tabela 7) prikazani obnašanje ter citati zalezovalcev, povzeti iz digitalnih dokazov, najdenih v preiskavi.

4.3 Metode AVR kot orodje v digitalni forenziki

Kombiniranje metod AVR s standardi v procesu digitalne forenzične preiskave se je v vseh dvajsetih primerih izkazala kot koristna v mnogih pogledih. V tem razdelku bomo na primerih pokazali koristnost te kombinirane metode.

4.3.1 Fokus, hitrost in smeri preiskave

V enem od primerov je žrtev prijavila sumljivo aktivnost na svojem prenosniku. Na ekranu so se ji redno pojavljala sporočila s sovražno vsebino. Za namen raziskave so bile najprej pregledane registrske datoteke žrtvinega prenosnika, ki so pokazale, da je bila nameščena zlonamerna programska oprema. To je pokazalo, da je bil žrtvin prenosnik ogrožen in so do njega dostopali iz daljave. Žrtev je imela v mislih dva osumljenca, s katerima je v preteklosti imela razmerje preko interneta. Ker žrtev ni imela zaščitenega prenosnika, oziroma sploh ni imela nameščenega antivirusnega programa, je zlonamerna koda z lahkoto prispela do njene naprave. Preiskava o tem, kako je bila nameščena zlonamerna koda na njenem prenosniku je potekala tako, da je bilo najprej izvedeno preiskovanje zlonamerne kode v vseh njenih datotekah. Rezultati so pokazali krajši seznam zlonamernih datotek in njihove logične lokacije na računalniku. Izstopala je prenesena datoteka, locirana v direktoriju klepetalnika, ki ga je žrtev uporabljala. Nadaljnja analiza je pokazala, da se z dvakratnim klikom na najdeno datoteko namesti zlonamerna koda, ki je bila predhodno odkrita. Z analizo beležk se je izkazalo, da je oseba, ki je bila ena od osumljencev, poslala zlonamerno kodo. Tako je žrtev nenamerno namestila zlonamerno programsko opremo na svoj

Tabela 7: Vedenje in motivacija spletnih zalezovalcev povzeta iz digitalnih dokazov

Verjetna motivacija spletnega zalezovalca	Dejanja/navedbe spletnega zalezovalca iz digitalnih dokazov
Lažne obtožbe žrtve	Objavljanje nespodobnih/spremenjenih slik žrtev na socialnih omrežjih. Pošiljanje nespodobnih/spremenjenih slik žrtev preko e-pošte prijateljem in svojcem žrtve. Objavljanje slik in osebnih informacij žrtve na spletnih straneh za zmenke. Pošiljanje žaljivih e-sporočil sodelavcem prek žrtvinega e-poštnega predala. Pošiljanje e-sporočil z nespodobnimi/lažnimi informacijami o žrtvi prijateljem/svojcem/sodelavcem.
Izpovedi ljubezni	Neprestano pošiljanje e-sporočil z izpovedmi ljubezni, ki kažejo obsedenost z žrtvijo. Neprestano pošiljanje e-sporočil o spominih iz preteklega razmerja. Pošiljanje e-sporočil, ki zahtevajo prekomerno pozornost. Pošiljanje osebnih/pornografskih slik.
Maščevalnost/jeza	"Do konca življenja boš obžalovala, kar si storila." "Kjerkoli si ... Našel te bom." [Ime izvršnega podjetja] je lažno, neprofesionalno in goljufa stranke."
Zbiranje informacij o žrtvi/Sledenje žrtvi	Dostopanje do žrtvinega računalnika na daljavo. Zbiranje informacij o žrtvi in organiziranje le-teh po datotekah.

prenosnik. Ostali zbrani dokazi so prav tako pokazali, da je zalezovalec oseba, ki je poslala zgoraj omenjeno datoteko.

Ta primer pokaže, kako lahko metode AVR pomagajo pri procesu preiskave z določitvijo specifičnega fokusa in smeri za strategije v nadaljnji preiskavi in določitvijo potencialnih iskalnih podmnožic. Zelo pomembno je razumeti kontekst napada, preden se lotimo analize potencialnih digitalnih dokazov. To namreč lahko pomaga preiskovalcu identificirati začetno lokacijo obnovitve dokazov, ne da bi preverjal in iskal dokaze na celotnem disku.

4.3.2 Vedenje in motivacije vpletenih

V drugem primeru je zalezovalec naredil lažni profil v imenu žrtve na spletni strani za zmenke. Zalezovalec je pri tem dodal nemoralne slike žrtve in objavljal sporočila, v katerih podrobno opisuje seksualne fantazije in opogumi ostale udeležence, da stopijo v stik z žrtvijo, z namenom intimnosti. Pri tem je objavil tudi njen e-poštni naslov. Žrtev je osumljenca prijavila policiji. Iz preiskave relevantnih dokumentov in žrtvinih aktivnosti na spletu je bilo razvidno, da je bila žrtev redna obiskovalka omenjene spletne strani. V intervjuju je žrtev izjavila, da je zalezovalca spoznala preko te strani ter da je imela z njim krajše razmerje, ki ga je končala. Preiskane so bile tudi beležke najnovejših pogovorov z zalezovalcem. Pogovori so se začeli z lepimi besedami, nakar se je njegovo obnašanje postopoma spremenilo v moteče obnašanje z intenzivnimi odzivi. Beležke so pokazale, da se je nekdo kar nekajkrat prijavil pod imenom žrtve. Na računalniku zalezovalca so bile kopije slik žrtve, ki jih je objavljal, očitno, da bi se maščeval za njuno propadlo razmerje. Ta primer v celoti pokaže, kako lahko digitalni dokazi zrcalijo sliko odnosa žrtev-storilec in tako pomagajo pri razumevanju dinamičnosti primera. V tem primeru so redni obiski žrtve na spletni strani za zmenke in srečevanje oseb, ki jih je spoznala preko spleta ali v živo, povečali tveganje, da postane žrtev. Vsi dokazi proti zalezovalcu nakazujejo, da se je maščeval za propadlo razmerje.

V skoraj vseh primerih je pisana spletna komunikacija ponudila dosti koristnih dokazov. Prepoznavni znaki v pisanju zalezovalca so lahko sintaksa, slovnične napake ter spletni nazivi. Kot koristno sredstvo forenzične preiskave so metode AVR ponudile dokaze glede razmerja vpletenih, pomagale pri razumevanju konteksta celotnega primera ter pomagale pri ocenjevanju najbolj relevantnega osumljenca. To je razvidno v tabeli (Tabela 7), iz prejšnjega razdelka.

Najbolj uporabne datoteke pri forenzičnem preiskovanju so uporabnikove osebne datoteke in direktoriji, kot tudi datoteke zgodovine brskanja. Iz njih lahko razberemo želje in navade žrtve, npr. iskani termini v brskalniku, pogosto obiskane spletne strani, ki so žrtev lahko izpostavili zalezovalcem. To so lahko kazalci interesov zalezovalca ali njegove motivacije. Na ta način se lahko najdejo relacije z drugimi potencialnimi žrtvami ali drugi dokazi, ki indicirajo, da se je zgodilo zalezovanje. Tako obstaja primer, kjer je storilec na računalniku hranil vse spremenjene slike žrtve skupaj z originalnimi slikami ter dokaze uporabe specialne programske opreme za ustvarjanje teh slik.

Glede na tip zalezovalca so lahko indikatorji obstoječih seksualnih interesov zalezovalca naslednji: število shranjenih datotek, njihova lokacija ter obstoj datotek, ki vsebujejo eksplicitni material. To lahko seveda predstavlja dodaten dokaz za motivacijo storilčevega vedenja. Na primer, obstoj otroške pornografije na disku osebe, osumljene zalezovanja mladoletnika, lahko pokaže, da je zalezovalec imel seksualno motivacijo za zalezovanje.

4.3.3 Identifikacija potencialnih žrtev

Če zalezovalec sortira ter kategorizira datoteke svoje žrtve, lahko sklepamo, da je zalezovalec res posvečen zalezovanju, saj si je vzel čas za sortiranje informacij. Tovrstna organiziranost lahko kaže, da obstaja več žrtev istega zalezovalca. V enem od primerov so preiskovalci na osumljenčevem računalniku naleteli na datoteko, ki je vsebovala več poddatotek, ki so vsebovale slike in informacije treh različnih žensk (ki so osumljenca tudi prijavile policiji). Storilec je datoteke verjetno uredil, da je lažje vodil evidenco svojih zalezovanj.

4.3.4 Eliminacija osumljencev

V zadnjem primeru, ki ga bomo predstavili, je zalezovalec oponašal žrtev na socialnem omrežju Facebook, kjer je objavljal njene slike skupaj z žaljivimi komentarji. Glavni osumljenec je bil njen bivši soprog. Z analizo njegovega prenosnika so bile pridobljene slike žrtve. Nadaljnja analiza zgodovine spletnega brskalnika je pokazala, da se je uporab-

nik prijavljal v žrtvin Facebook profil iz tega računalnika. Vse je kazalo na to, da je zalezovalec bivši soprog. Po drugi strani pa je časovna analiza pokazala največjo aktivnost na omenjenem Facebook profilu vsak delavnik od 9h do 14h. Izkazalo se je, da je bil osumljenec v tem času vedno v službi. Edina druga oseba, ki je imela dostop do računalnika je bila osumljenčeva nova žena, ki je bila ob omenjenem času vedno doma.

Zato je pri izvedbi forenzične analize vedno pomembno izvesti časovno analizo. Dokazno gradivo se ne sme obdelovati posamično. V kolikor je mogoče, preiskovalci poskušajo podatke o datumih in časih pobranih podatkov navezati z drugimi časovnimi žigi (iz izjav žrtve ter zalezovalca). Tako lahko izdelamo časovnico za vse aktivnosti zalezovalca in žrtve, ki lahko pomaga pri rekonstrukciji primera. Po drugi strani pa pridobimo časovni okvir aktivnosti, ki se lahko preverijo s tistimi, ki jih žrtev omenja. Najbolj pogosto spremembe v podatkih o časovnem žigu datoteke pokažejo spreminjanje datoteke (na primer ali je slika bila spreminjana v sliko z neprimerno vsebino). Časovni žigi ali zastavice lahko pokažejo tudi koliko časa je zalezovalec imel te datoteke in planiran čas trajanja zalezovanja pri določeni žrtvi. To pomeni, da korelacija časovnih žigov datoteke z dnevno aktivnostjo osumljenca lahko pomaga določiti potencialnega zalezovalca v primerih, ko različni ljudje uporabljajo en računalnik.

5. ZAKLJUČEK

Spletni zalezovalec lahko deluje v več oblikah: z lažnimi obtožbami (škodovanjem ugleda, širjenjem lažnih novic), z zbiranjem osebnih informacij, z opazovanjem žrtvenih spletnih aktivnosti, s hujskanjem drugih proti žrtvi, z napadanjem na podatke in opremo s škodljivimi programi, z naročanjem dobrin in storitev v imenu žrtve. Pogosti so tudi primeri, ko se zalezovalec pretvarja, da je sam žrtev [1]. Veliko ljudi je ranljivih na spletno zalezovanje, ker ne upoštevajo preventivnih ukrepov kot so redno posodabljanje protivirusnih in protivohunskih programov ter uporaba požarnega zidu. Pogosto delijo osebne informacije na profilih in javnih mestih, uporabljajo šibka gesla, jih ne zamenjujejo redno ali uporabljajo enaka gesla za več različnih računov. Paziti je treba na nenadne spremembe na mobilni napravi - na primer nenadna velika poraba interneta ali baterije lahko pomeni, da je bil naložen vohunski program, lahko nekdo sledi preko GPS-a, bere sporočila ... Preko 50% bivših partnerjev začne z zalezovanjem še preden se sploh razidejo - znaki so vidni tudi v obnašanju in sicer v pretiranem nadzorovanju, klicanju in povečani agresivnosti [1].

Z deduktivnimi metodami značilnosti storilca razberemo iz vsakega primera posebej. Prednost pri raziskovanju primerov spletnega zalezovanja so ravno podatki, ki se lahko pridobijo iz storilčeve naprave. Preden se analizirajo digitalni dokazi, je treba razumeti kontekst vsakega primera posebej. Zelo pomembna je časovna analiza. Z metodami AVR preko štirih korakov iz forenzičnih dokazov in samega kaznivega dejanja določamo lastnosti storilca kaznivega dejanja in motivacijo. Na ta način lahko prepoznamo možne spremembe v storilčevem razmišljanju in obnašanju, ki jih s statistiko ne bi. Lažje ugotovimo primere, ki se s karakteristikami splošne statistike ne skladajo. V primerih v raziskavi na Dubajski policiji [17] so vse žrtve poznale zasledovalca, kar je v nasprotju s statistiko. Prav tako je v nasprotju, da prevladujoče prejšnje razmerje ni ljubezensko (največ je bilo sodelavcev).

Omenili smo tudi problematiko zakonodaje na področju spletnega zalezovanja. Odkrili smo, da je bil letos (januarja 2016) dodan nov 134.a člen Zalezovanje, ki pravi: *Kdor koga drugega ali njegovega bližnjega s ponavljajočim se opazovanjem, zasledovanjem ali vsiljivim prizadevanjem vzpostavitve neposrednega stika ali stika preko elektronskih komunikacijskih sredstev zalezuje in pri njem ali pri njegovem bližnjem s tem povzroči prestrašenost ali ogroženost, se kaznuje z denarno kaznijo ali zaporom do dveh let. Če je zalezovana oseba mladoletna oseba ali slabotna oseba, se storilec kaznuje z denarno kaznijo ali z zaporom do treh let.*[1]. Ključni členi kazenskega zakonika so z vidika spletnega zalezovanja prej bili: nedovoljena objava zasebnih pisanj (140. člen), zloraba osebnih podatkov (143. člen) in kršitev moralnih avtorskih pravic (147. člen) [13].

6. LITERATURA IN VIRI

[1] Cyberstalking. `http://www.digital-trust.org/stalking`. [Dostopano: 12-Maj-2016].

[2] E. M. Alexy, A. W. Burgess, and T. B. amd Shirley A. Smoyak. Perceptions of cyberstalking among college students. 2005. [Dostopano: 11-Maj-2016].

[3] J. Ashcroft. Stalking and domestic violence: Report to congress. 2001. [Dostopano: 11-Maj-2016].

[4] K. Baum, S. Catalano, and M. Rand. Stalking victimization in the united states. 2009. [Dostopano: 11-Maj-2016].

[5] P. Bocij and L. McFarlane. Online harassment: towards a definition of cyberstalking. 2002.

[6] J. Bryce and J. Fraser. The role of disclosure of personal information in the evaluation of risk and trust in young peoples' online interactions. 2014. [Dostopano 11-Maj-2016].

[7] E. Casey. *Digital Evidence and Computer Crime, Second Edition*. Academic Press, 2004. [Dostopano: 1-Maj-2016].

[8] FRA - European Union Agency for Fundamental Rights. *Violence against women: an EU-wide survey*. FRA, 2014. [Dostopano: 1-Maj-2016].

[9] B. Gaille. 20 provocative cyberstalking statistics. `http://brandongaille.com/20-provocative-cyberstalking-statistics/`, 2014. [Dostopano: 8-Maj-2016].

[10] D. Garlicki. Ex-trooper gets prison for nudes. `http://articles.mcall.com/2007-09-07/news/3767707_1_ex-wife-harassment-state-trooper-raymond-judge/`. [Dostopano: 11-Maj-2016].

[11] J. D. Harvey. Cyberstalking and internet harassment: What the law can do. 2003. [Dostopano: 8-Maj-2016].

[12] H. Jahankhani and A. Al-Nemrat. Examination of cyber-criminal behaviour. 2012. [Dostopano: 11-Maj-2016].

[13] T. Janežič. Kibernetsko nadlegovanje in zalezovanje otrok. magistrsko delo. `https://dk.um.si/Dokument.php?id=71059`, 2015.

[1]Kazenski zakonik (KZ-1-NPB4), 134.a člen: https://zakonodaja.com/zakon/kz-1/134a-clen-zalezovanje

[Dostopano: 8-Maj-2016].

[14] R. M. Kowalski, L. S. P., and A. P. W. Cyberbullying: Bullying in the digital age. 2012.

[15] T. E. McEwan, P. E. Mullen, and R. MacKenzie. A study of the predictors of persistence in stalking situations. 2008.

[16] L. McFarlane and P. Bocij. An exploration of predatory behaviour in cyberspace: Towards a typology of cyberstalkers. 2003. [Dostopano: 8-Maj-2016].

[17] N. A. Mutawa, J. Bryce, V. N. L. Franqueira, and A. Marrington. Forensic investigation of cyberstalking cases using behavioural evidence analysis. 2016.

[18] N. Nykodym, R. Taylor, and J. Vilela. Criminal profiling and insider cyber crime. 2005.

[19] M. K. Rogers. The role of criminal profiling in the computer forensics process. 2003.

[20] M. K. Rogers. Psychological profiling as an investigative tool for digital forensics. 2015.

[21] A. Silde. Profiling the cyberstalker. `http://commerce3.derby.ac.uk/ojs/index.php/da/article/download/67/60`, 2014. [Dostopano: 9-Maj-2016].

[22] A. Silde and O. Angelopoulou. A digital forensics profiling methodology for the cyberstalker. 2014.

[23] Statista. Most common ways in which cyber stalking cases escalated in 2013. `http://www.statista.com/statistics/291257/cyber-stalking-victims-how-case-escalated/`. [Dostopano: 11-Maj-2016].

[24] P. R. Stephenson and R. D. Walter. Toward cyber crime assessment: cyberstalking. 2011. [Dostopano: 11-Maj-2016].

[25] B. E. Turvey. *Criminal Profiling, Fourth Edition: An Introduction to Behavioral Evidence Analysis, 4th Edition*. Academic Press, 2011.

[26] WHOA. Whoa 2013 cyberstalking statistics. `http://www.haltabuse.org/resources/stats/2013Statistics.pdf`, 2014. [Dostopano: 11-Maj-2016].

[27] Wikipedia. Wikipedia-matthew pyke. `https://en.wikipedia.org/wiki/Matthew_Pyke/`. [Dostopano: 11-Maj-2016].

[28] S. Yu. Behavioural evidence analysis on facebook: A test of cyber-profiling. 2013.