Univerza v Ljubljani

Fakulteta za računalništvo in informatiko

University of Ljubljana

Faculty of Computer and Information Science

Zbornik

Digitalna forenzika

Seminarske naloge, 2020/2021

Ljubljana, 2021

Zbornik

Digitalna forenzika, Seminarske naloge 2020/2021

Editors: Andrej Brodnik, študenti

Template author: David Klemenc

Ljubljana : Univerza v Ljubljani, Fakulteta za računalništvo in informatiko 2021.

Kazalo / Contents

1 Uvod / Introduction

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

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

V zborniku so zbrane seminarske naloge študentov magistrskega študija na Fakulteti za računalništvo in informatiko, Univerze v Ljubljani 2020/2021. V okviru predmeta Digitalna forenzika je vsaka skupina študentov izbrala en članek, ki je služil kot izhodišče za seminarsko delo. Članki so bili izbrani iz treh raziskovalnih področij: forenzika v kazenskem postopku, forenzika slik in forenzika metapodatkov, na koncu pa sledijo članki, ki ne spadajo v nobeno od prej omenjenih kategorij.

Na področju forenzike v kazenskem postopku se seminarske naloge osredotočajo na zajem in obdelavo digitalnih dokazov. Bolj natančno kakšno vlogo imajo IP naslovi kot digitalni dokazi in kako jih obravnavamo v sodnem postopku. Prav tako je prikazan aktualen primer takega postopka, ki se je zgodil na Sloveškem.

Forenzika slik je dandanes zelo pomembna, saj postaja manipulacija s slikovnimi datoteki vedno bolj preprosta in s tem tudi vedno bolj razširjena. Seminarske naloge se v tem delu ukvarjajo z iskanjem informacij o izvoru neke slike in zaznavanjem Deepfake videev, predvsem pa se osredotočajo na zaznavanje raznih manipulacij in ponarejanjem slik.

Forenzika metapodatkov se ukvarja z iskanjem in zbiranjem informacij o datotekah na računalnikih, spletu, itd. V seminarskih nalogah so prikazani primeri iskanja datotek na neznanih medijih in preiskovanje informacij o slikah glede na priložene metapodatke.

V sklopu razno pa so seminarske naloge, ki ne spadajo pod noben omenjen sklop. Te so se ukvarjale z napadi preko gruč pamethinh naprav, analizo mobilnih naprav in analizo anonimnega brskalnike Tor ter njegovega omrežja.

Ta zbornik združuje vse končne seminarske naloge, ki so bile izdelane v študijskem letu 2020/2021. Namenjen je vsem, ki jih zanima področje digitalne forenzike ali pa zgolj eno ali več predstavljenih področij.

Ljubljana, 2021

2 Povzetki / Summaries

2.1 IP Addresses in the Context of Digital Evidence in the Criminal and Civil Case Law of the Slovak Republic

With the world becoming increasingly digital and the increasing amount of information being transferred through networks around the world, it is also important to take into account the digital evidence in crimes. One of the parts of the digital evidence is also an IP address which was logged. In this essay we take a look at how IP addresses are used in courts in Europe and whether IP address is considered as a personal information. We find that not all courts agree in regards to this aspect and that there are several contradicting court decisions in the past 20 years. Besides that we also describe methodology which was used by Slovak researchers to analyze the use of IP addresses in their courts and report their findings on relationship between various factors and their importance on the final decision of the court and juridical practice regarding the anonymisation of IP addresses in court decisions. We also present the results our own research in which we examined some recent court decisions in Slovenia regarding the use of IP addresses and user privacy.

2.2 DeepUAge: Improving Underage Age Estimation Accuracy to Aid CSEM Investigation

Age estimation models are useful for identifying possible materials involved in Child Sexual Exploitation Material (CSEM) distribution. These models are build using deep learning networks. They are trained over specific datasets. Working with datasets, especially for such cases, is a challenge on its own due to the fact that much of the data isn't labelled or is mislabeled. Although the accuracy of similar models has been improved over the years, there are still challenges they face, some of which will be mentioned in this paper. In this paper, we go back to the basis of building such model, highlight the features and jump onto challenges it faces.

2.3 Prepoznavanje deepfake posnetkov z odkrivanjem obraznih artefaktov

V seminarski nalogi opisujemo odkrivanje globokih ponaredkov (angl. deepfake) na podlagi odkrivanja artefaktov, ki nastanejo pri ponarejanju posnetka. Zato bomo najprej naredili pregled področja in ugotovili, kakšne tehnike se danes uporabljajo za odkrivanje globokih ponaredkov. Nato bomo opisali, kaj je globok ponaredek, ter kakšne vrste artefaktov lahko nastanejo pri delanju globokih ponaredkov. Na koncu pa si bomo pogledali implementacijo modela za zaznavanje globokih ponaredkov na podlagi odkrivanja artefaktov, ki je podrobneje opisana v članku.

2.4 On Learning to Detect Manipulated Facial Images

The paper describes the realism of image manipulation, and how difficult is to detect them automatically or by humans. With the rapid development of image generation and manipulation there is a need for a system to detect manipulated images. Manipulated images can be used for spreading false information, identity stealing etc. They created automated benchmark for facial manipulation detection for easy comparing.

2.5 Generični algoritem za klesanje metapodatkov na podlagi časovnih žigov

Na področju digitalne forenzike ima veliko vlogo klesanje datotek. To je postopek pridobivanja datotek iz medija, o katerem nimamo podatkov, kje se datoteke nahajajo. Taki postopki so več ali manj uspešni pri pridobivanju samih datotek, redko pa se osredotočajo na metapodatke, povezane z njimi. Avtorji članka so razvili nov generični algoritem za klesanje metapodatkov na podlagi časovnih žigov in ga testirali na datotečnem sistemu NTFS in ext4. Algoritem se je dobro izkazal, saj je našel vse datoteke na poškodovanem NTFS disku.

2.6 Searching for the source of images based on information in image headers

While there are many different approaches to trace the provenance of an image many of them are demanding in resources and complex to build, which makes them costly to apply to individual cases of digital forensic. We investigate an approach of grouping JPEG images by their file headers, which is already considered as lightweight, and try to simplify it even more to make applying a machine-learning model feasible in everyday digital forensic investigations while still reaching applicable results. We refer to the ideas of P. Mullan et. al. (2020), stripping their models from even more complexity unifying it to one model. Our findings (overall accuracy 98%) support that the path of building lightweight machine-learning models for grouping images by provenance is actually promising as suggested in the original paper.

2.7 IoT Botnet Forensics

Botnets are a relatively new thing in the world of IoT. Since the attack vectors now range from smart refrigerators to security cameras instead of just different types of operating systems we don't yet know the best approaches to tackle these infections forensically. Despite the low performance of these kinds of devices they are widely spread and can cause serious outages due to their numbers. Additionally we learn to protect our computers using antivirus software, installing updates, etc. we usually never do that with IoT devices which decreases the chance of mitigating a botnet's spread. In the year 2016 the Mirai botnet was activated and it took down more than a 1000 server which included widely used sites like Twitter, Netflix and DynDNS as shown in figure 1. And with the release of its source code, even more botnets were created which started using these devices for different malicious purposes. This study is focused on this exact botnet and its variants that came from the released source code to help the forensic researcher's to discover, monitor and act on active new implementations. The components of the botnet, like the Command and Control (CNC) server, Scanner and Loader, are described in detail from a forensic point of view. The descriptions contain different data collection methods, parsing the information from the data and breaking down the core parts of the code and network traffic which can be used for obtaining useful information about new similar botnets and their mitigation.

2.8 Parse and validate mobile forensic artifacts with Python

Mobile forensics is becoming more and more important as they are more and more present in our daily life. The data collected by our smartphones are a great help for investigators. However, it can be cryptic at first, which is why many researchers are working to understand what data is stored and how. To exploit this data it is necessary to build tools to visualize this data correctly and to make an efficient analysis. In this paper we will see how we can have access to the last image of an application, data allowing the construction of a timeline of the phone use. Finally, we will show two tools to exploit the databases present in the phones.

2.9 Forenzična analiza brskalnika Tor

V današnjem času varnost in anonimnost v spletu postajata vedno bolj relevantni. Uporabniki skušajo ostati anonimni z uporabo brskalnikov, kot je Tor. Njegov namen je zakriti uporabnikovo lokacijo in dejanja pred analizo prometa v internetnem omrežju. Uporaba Tora pa po drugi strani forenzikom oteži sledenje kriminalnih internetnih aktivnosti. V analiziranem članku so se avtorji osredotočili na forenzične artefakte, ki jih za seboj pusti uporaba Tora in ne toliko na spletne aktivnosti, katerim je primarno namenjen. V članku najprej opišemo omrežje Tor in kako je le-to realizirano. Nato opišemo še brskalnik in podatke, ki jih Tor shranjuje. Zatem na kratko navedemo rezultate obstoječih raziskav. V nadaljevanju podamo metodologijo, uporabljeno v analiziranem članku, rezultate, ki so jih raziskovalci dosegli in glavne ugotovitve, do katerih so prišli. Povzamemo, da sta omrežje in brskalnik Tor namenjena predvsem omogočanju anonimnosti uporabnika v spletu, ne pa tudi zakrivanja uporabe brskalnika v osebnih napravah. Predlagamo, da bi bila izvedba podobne raziskave koristna tudi na sistemih Unix.

IP Addresses as a Digital Evidence in the Criminal and Civil Case Law

Tim Štromajer
Faculty of Computer and Information Science
Večna pot 113
Ljubljana, Slovenia
ts3302@student.uni-lj.si

Miha Kosi
Faculty of Computer and Information Science
Večna pot 113
Ljubljana, Slovenia
mk9930@student.uni-lj.si

Nik Zupančič
Faculty of Computer and Information Science
Večna pot 113
Ljubljana, Slovenia
nz7793@student.uni-lj.si

ABSTRACT

With the world becoming increasingly digital and the increasing amount of information being transferred through networks around the world, it is also important to take into account the digital evidence in crimes. One of the parts of the digital evidence is also an IP address which was logged. In this essay we take a look at how IP addresses are used in courts in Europe and whether IP address is considered as a personal information. We find that not all courts agree in regards to this aspect and that there are several contradicting court decisions in the past 20 years. Besides that we also describe methodology which was used by Slovak researchers to analyze the use of IP addresses in their courts and report their findings on relationship between various factors and their importance on the final decision of the court and juridical practice regarding the anonymisation of IP addresses in court decisions. We also present the results our own research in which we examined some recent court decisions in Slovenia regarding the use of IP addresses and user privacy.

Keywords

Digital forensics, digital evidence, IP address, cybercrime, investigation, privacy

1. INTRODUCTION

During the last year pandemic forced people around the world to rely on the internet for many of their daily tasks: working from home, studying, banking, buying groceries and keeping in touch with friends and relatives are just a few of the things we use on a daily basis. However rising amount of internet traffic also introduces new opportunities for digital criminals around the world with consequences that can be just as damaging to the victim as physical crimes. When it comes to digital crime, digital data is often the only evidence the investigators have available at first but digital evidence is not only limited to digital crimes. We carry our smart devices with us every day and collect even more data than we are aware of, which can be an important source of information for investigators. Computers and other digital devices can be used in various ways: as a means of communication between perpetrators, as a tool to gain information about the victim or even carry out the criminal act using the computer. Some of the questions that the digital evidence can answer are: Who was the suspect coordinating the crime with? Were the illegal items or substances obtained through the internet? What kind of activities did the suspect engage in before and after the crime was committed?

For that kind of evidence to be used properly during prosecution both lawyers and judges have to understand the evidence that is obtained. The role of digital forensics in a criminal investigation is to obtain, analyse and preserve digital evidence. The goal of this essay is to compare how IP addresses are used in criminal cases around the world, explore how computer's IP address is treated when presented as an evidence in the court specifically in case of Slovak Republic and also research the lawfulness of user identification using IP addresses in Slovenia.

In chapter 2 we will review some basics about network and network security and how IP addresses are assigned, in chapter 3 we take a look at some related work and how courts across Europe prosecute cybercrimes and treat IP addresses that were collected. In chapter 4 we focus on our main paper focusing on Slovak courts [20] we describe how the authors of the original article obtained and prepared data for their research and chapter 5 describes the methods of analysis, the findings of the authors of the original article and our research and results. Finally in chapter 6 we sum up the results and present the solutions to some of identified issues proposed by the authors of the original article and also our own.

2. NETWORKS AND NETWORK SECURITY

The Internet started out as a military communication system designed to withstand attacks and enable communication around the world - network called ARPANET has been designed in 1969 and is considered a foundation of modern Internet [7]. It became widely available to the public with the invention of World Wide Web in 1991. Since then it has grown significantly and became an important part of our lives. Just about every electronic device today has the capability of communicating wirelessly - mobile phones, watches, televisions and other household appliances.

A computer network is a system of multiple devices in which every pair of devices can communicate [14]. Devices connected to the Internet can communicate between each other because they speak a common language: TCP/IP. It is a set of communication protocols which specify how data should be addressed, transmitted, routed and received [6]. These tasks are organised into 4 layers (from bottom to top):

- **Link layer:** contains communication methods for data within single network segment

- **Internet layer:** provides connection between independent networks
- **Transport layer:** handles host to host communication
- **Application layer:** provides process to process communication

Every device on a network has its own address called **Internet Protocol address** or **IP address** for short and serves two main purposes: host identification and location addressing. Internet Protocol version 4 (IPv4) defines an IP address as a 32 bit number but because of the rising number of devices and lack of unique IP addresses IPv6 was established where IP address is a 128 bit number and offers a much larger address space. The Internet service provider (ISP) can assign a dynamic or static IP address to the user's network [7]. A dynamic address means that the provider can assign a new IP address to the customer whenever necessary. This allows a more efficient use of available addresses to the ISP and is becoming the norm for many providers around the world. Static address on the other hand means that your IP does not change and it has its own use cases, for example if you were hosting a website you would want a static IP address so your website would always be available on the same address. From the forensic standpoint dynamic addresses could be problematic in identifying who was using a certain IP address but providers log the assignments of dynamic IP addresses.

Relying on IP address to identify the perpetrator does not always give accurate information. Virtual private network (VPN) extends local network by providing an encrypted tunnel to that network [7]. It is especially useful for employees working outside of their offices as VPN allows them to connect to the corporate network even when working from home for example. However in the recent years companies offering VPN services to servers all around the world are becoming more and more popular, promising the users more privacy and ability to hide their real location from the advertisers. While the security and traceability of VPN is not our main point of research it is important to point out that in forensics tracing an IP address that was used does not necessarily give us the actual location where the perpetrator was at the time of a crime and that their real address could be masked with a different IP from a different country.

3. EVIDENCE IN (CYBER)CRIMES

When investigating a crime the goal of the investigators is to obtain as much evidence as possible to get a clear picture on what happened, how it happened and more importantly who committed the criminal act. Digital evidence is defined as *any data stored or transmitted using a computer that support or refute a theory of how an offence occurred or that address critical elements of the offence such as intent or alibi* [7], though this type of evidence is not only limited to cybercrimes. With how widespread the technology is today almost every person uses some kind of a smart device which means we inevitably leave traces of our activity: online messaging services, e-mail communication, websites we visit, applications we use and much more. Suspects may be able to cover up physical evidence of their crimes but with digital evidence this becomes virtually impossible. At the very least a web server will log every request it receives which contains: time of the request, IP address from which the request was sent, type of request and path to the requested resource. We are interested in exploring how much information can an IP address provide to the investigators and how it can be used in court.

3.1 Explanatory analysis of legal documents

In the past, countries have tried to develop conventions, to pursue a common criminal policy. The first such international treaty seeking to harmonise national laws, to improve investigation techniques in individual states and to improve cooperation between these states when investigating and prosecuting cybercrime is the Budapest Convention on Cybercrime in 2001. As of December 2020, 65 states have ratified the convention, while a further four states had signed the convention but not ratified it. It is a treaty on crimes committed via the Internet and other computer networks, dealing particularly with infringements of copyright, computer-related fraud, child pornography, hate crimes, and violations of network security.

Apart from national laws, there are many studies which focus on explaining the specifics of legal requirements in different countries. Comparative studies on the other hand are quite rare. One such example is Council of Europe's European Committee on Legal Co-operation [9], which is the Council of Europe intergovernmental body responsible for the standard-setting activities of the Council of Europe with a wide scope of competence in the field of public and private law. Its main role is to draw up standards commonly accepted by the 47 member states and to foster legal co-operation among them.

There is a small amount of studies focusing on less significant countries that go beyond the mere description of legal requirements or small-scale comparative notes. The most attention of international scholars and their readership is often dedicated to legal requirements in the USA and the United Kingdom. Paper from Montasari [18] explores how the admissibility of digital evidence is governed within the United Kingdom jurisdictions. In paper from Cole et al.[8] they explore the different issues seen in the prosecution of digital forensic investigations. 100 appellate cases were randomly selected using the FindLaw database, which is a database of case law from the U.S. Supreme Court and U.S. Circuit Courts of Appeal, as well as several state supreme courts.

3.2 Is IP address personal data?

The question of whether IP addresses should be treated as personal data is not new and has already been discussed in several published papers. Norwegian attorney at law Patrick Lundevall-Unger and researcher Tommy Tranvik published a paper in 2011 titled *IP addresses - Just a number?* [16] in which they provide several court rulings in European courts where IP addresses were used:

- In a court case which happened in Munich in 2008 the plaintiff claimed that a website violated the German Data Protection Act because it was storing the dynamic IP address and timestamp of the visit which could expose the identity of the visitor. The court

ruled in favour of the defendant claiming that the website operator can not link IP address to user's identity using 'normally available tools' (that is without using disproportionate amount of resources, illegal means or help of a third party)

- Members of a file sharing network appealed to the ruling of court in Paris in 2007, claiming that processing of their addresses should be authorised by the French Data Protection Authority. Their appeal was denied by the court with a similar explanation to the case in Munich stating that only legitimate authority may obtain the identity of a user from Internet Service Provider and that illegal methods of connecting user to his address should not be taken into account.

Both courts reached a similar conclusion with regards to not taking into account the illegal methods of unmasking the IP addresses when deciding whether IP address is personal data but not all courts agree with that:

- Similar to Munich court case, the plaintiff argued in a case that a website was violating German Data Protection Act by logging the IP addresses. Regional and District Court in Berlin both decided that all means of identification from IP should be taken into account regardless of who controls them and therefore IP addresses are indeed private data.
- Few years earlier (in 2005) the Stockholm court similarly argued that website provider could obtain visitor's information by contacting a third party, e.g. Internet service Provider. Because of that the court again decided that IP address should be treated as personal data.

The latter cases both directly contradict the former description and disregard whether the methods of obtaining identity from data are illegal but instead only focus on how much effort or the required cost needed to link IP address to name and face of a visitor. The paper [16] also proposes two step interpretation of European Union's Data Protection Directive (predecessor of GDPR) which consists of:

1. **Legality test** - identify and exclude illegal methods to linking names and faces to IP addresses.
2. **Likely reasonable test** - evaluation of effort and costs for various means of identification, including illegal methods. The larger the cost and effort needed, the less likely it is that IP address will be treated as personal data.

European Union adopted the **General Data Protection Regulation** in 2016 which among other things also defines the term **personal data** as *any information related to an identified or identifiable natural person.* Recital 30 of GDPR mentions online identifiers such as IP address, cookie identifiers or RFID tags which could be combined with other identifiers to create profiles of natural person and identify them. Identifying a person under GDPR does not necessarily mean linking the collected data to names or faces but it's enough that we can distinguish an individual from other individuals solely by looking at the data. IP address under GDPR is considered as personal data.

Due to many variables, such as if an IP address is dynamic or static or IPv4 or IPv6, however, there have been mixed opinions whether just an IP address as a whole is considered a personal data. The most popular opinion is that static IP addresses are considered personal data, when they can be connected to natural individuals, which is mostly due to the possibility of identifying them using WHOIS requests. This opinion also directly implies that IPv6 IP addresses are also considered personal data, because in this case every device receives a single address. In general, however, some legal scholars think that IP addresses should be considered as personal data, as long as it is theoretically possible to identify an individual, while some think that IP addresses should only be considered as personal data, if the data controller is able to identify the individuals using the addresses [23].

3.3 Prosecution of cybercrimes

In 2019 two Czech researchers took on an interesting question [10]: which crimes relating to attacks on integrity, accessibility and confidentiality of data are actually prosecuted in Czech courts. While this research focuses solely on the cases in Czech Republic it still gives us an insight on trends of cybercrime in Europe and how successful local law enforcement agencies are in dealing with it. To analyse this question they obtained all cases involving illegal access, data interference and system interference (articles 2, 4 and 5 of the Convention on Cybercrime [13]) between 2008 and 2016 in Czech Republic. They concluded that only the 'simple' cases of cybercrime and even then only a small number of those are prosecuted while more complex acts of cybercrime are usually not prosecuted. Another fact that researchers emphasised is that most of the cybercrimes that were prosecuted were between individuals who already knew each other on a personal level before the attacks took place (work colleagues, family members, spouses and partners) while the cases where the attacker and victim did not know each other were rare. The most noticeable trend is the increase of password misuse/unauthorised login with the goal of gaining access to private information, erotic materials, financial profit or taking revenge against the victims.

Authors also suggested that Czech criminal procedure was not capable of dealing with more sophisticated acts of cybercrime: e.g. crimes which were carried out from abroad against the victims in Czech Republic or vice versa - attacks carried out from Czech Republic across the borders. The paper concludes that those acts of cybercrime brought to court are only a small fraction and not representative of the cybercrime in general. Many crimes were not prosecuted either because victims did not report them or because law enforcement simply did not have the means or knowledge to find the offenders.

To counter these issues Czech police set up a country wide department which among other sections includes a section for cybercrime. The goal of this department is to focus on more sophisticated cyber attacks with serious consequences for the victims. Besides that a police hotline was established for reporting any suspicious content or activity found on the

internet.

3.4 Internet privacy in United States

We have also taken a look at how courts in United States treat the IP addresses and other private data on the Internet. In general, federal courts agree that once users enter the cyber domain the existing privacy rights do not apply anymore [15]. A Supreme Court decision from 1967 [12] determined the expectation of privacy in two parts: a user must have a subjective expectation of privacy and society must view that expectation as reasonable. This two step approach is still used by courts today when evaluating privacy matters. The general agreement among the courts in United States is that user's Internet activity (including IP address) is not a private activity and users should have no expectation of privacy [21]. Additionally if criminals use Tor browser which uses a concept of routing the requests through different proxies, an IP address is still provided to the network however it does not reveal the origin of those requests. Even in this case the courts do not believe that IP address is private information as the user must still disclose his own IP address in order to use the network [22]. While user does take steps to conceal his personal address, he still has to provide that personal information to a third party.

The IP addresses found during an investigation can be used to obtain a warrant but it does not necessarily lead to the right person [15]. Investigators can demand information about the owner of the router using that address but the person committing a crime is usually not the owner. One extreme case was the investigation of downloading child pornography on an unsecured network. The owner of the network faced an 18 month investigation because the investigators failed to mention that network was unsecured and anyone in the vicinity could use it [11]. Cases like this illustrate the importance of expert testimony and how important it is that all aspects of collected evidence are taken into account when being presented to the court. On the other hand it is also clear that only IP address and the ISP data tied to that address is not enough to narrow down a list of suspects and additional evidence should be provided to avoid investigation of innocent people which often causes unwanted consequences to those being investigated: damage to the reputation in local community, loss of trust from others, job loss etc.

With the criminals becoming more advanced the investigators have been using new (and questionable) techniques. In a case of child pornography website hosted on Tor network, the FBI agents have received information about the IP address of the host which allowed them to use that host's connection and distribute malware to the users who downloaded the content from the website. Despite many privacy concerns being raised, the federal courts across the United States agreed that this was a legal method of obtaining the user's information. These kind of techniques may be useful as criminals are often a step ahead of the law enforcement when it comes to technological knowledge however this sets a dangerous precedent in regards to how much power an investigator holds.

In general treatment of IP addresses in American courts is an interesting contradiction to European courts as in the case of United States courts all agree that there should be no expectation of address' privacy. In regards to how much information the IP address itself really carries the author [15] concludes that it can be a good starting point for the investigation however judges should keep in mind that an 'owner' of that IP address is not necessarily the user who committed a crime and additional evidence is needed to determine the suspect such as: protection of the network and other people using the network at the same time.

4. METHODOLOGY OF RESEARCH

For the purpose of researching how IP address is used as digital evidence in court [20], the authors of the paper first had to obtain all court decisions of Slovak courts between the years 2008 to 2019. This includes verdicts from courts of first instance, second instance, Supreme Court and Constitutional Court. They then filtered out only those that contain the term 'IP address' (398 verdicts) and additionally also removed those verdicts where the term IP address was mentioned but was not used as evidence. The final analysis contained 187 court decisions.

First question the researchers tried to answer was: *How is IP address used as evidence in court and what attributes are important for the court's decision?* To answer this question they collected the following data from each court decision:

- Is timestamp provided for IP addresses?
- Was IP address treated as a connection to a person, device, both or none?
- Did the document include further specification of device tied to IP address?
- Was a person's relation to the device considered?
- Did the court decision consider the possibility of other people accessing the device in question?
- Is it a personal or a work related device or other?
- Was the mention of an IP address followed by additional data relating to the address?
- Which evidence was considered in the case (defendant testimonial, witness testimonial, documentary evidence, real evidence or expert opinions)?

The second question researchers tried to answer in this paper is: *How do courts approach anonymization of IP addresses in their documents?* For this purpose they required different attributes:

- Was traditional personal data (such as name, surname, date of birth) anonymized?
- Were other online identifiers (username, mail address) anonymized?
- Was IP address anonymized?
- Were the IP addresses static or dynamic?
- Were the IP addresses private or public?

- Were the IP addresses of version 4 or version 6?

To allow further analysis, attributes were transformed into binary values even in case of 'non-binary' answers. For example, the attribute on which evidence was considered in the case would be transformed into multiple binary attributes: was defendant testimonial considered, was witness testimonial considered and so on. This way every attribute only has a yes or no value.

Researchers [20] first took a look at the dependence between variables using a measure called correlation coefficient. A high, positive coefficient between two variables would indicate that a higher value of one variable would indicate a higher value of the other variable while a negative correlation coefficient means that a higher value of one variable would correlate to a low value of the other variable. On the other hand, correlation coefficient close to zero would mean that the two variables in question are not related.

The next step of research uses logistic regression analysis [17] to explore how different types of evidence influence the court's decision and whether certain attributes are more significant than other. Logistic regression uses logistic model which estimates the probability of the output given some input.

Final stage consists of identifying specific groups of cases by using hierarchical clustering method [19]. This is a type of a clustering method which recursively partitions the instances in top-down or bottom-up fashion. The result of the process then needs to be manually evaluated using visual inspection to determine the number of clusters. Obtained clusters (groups of cases) are then interpreted to determine what the cases within a single clusters have in common.

5. RESULTS AND DISCUSSION

In this section, the results of the authors' and our research are presented. We take a closer look at both the relationship between the different attributes and between the individual attributes and the outcome of the decision. Then, we present the authors' findings on how the personal data is protected in court decisions in the Slovak Republic and the spacial analysis of IP addresses. Finally, we present our findings on the use and privacy of IP addresses in Slovenia, which represents our addition to the topic.

5.1 Correlation analysis of attributes

The authors created a heat map in figure 1 with correlation coefficients for all pairs of attributes using Pearson correlation. It shows the relationship between the individual attributes as presented in the methodology. We can clearly see direct dependency between expert opinion and other uses of particular types of evidence. This means, for example, that if an expert opinion was considered in the judgement's reasoning, there is a high probability that other types of evidence were also considered by the court. This includes real evidence, documentary evidence, testimony of a witness and testimony of a defendant.

Another dependency also exists between expert opinion and person's relation to the device. This illustrates that this requirement may have firstly been researched in the expert opinion itself.

High correlation coefficient, with a value of 0.56, also exists between the assignment of an IP address to a device and the specification of the device. This tells us that if the court correctly links an IP address to a device and not to a person, it is more likely that the device considered will be identified and further defined in the decision.

Specification of the device is also highly correlated to person's relation to the device. This relationship may suggest the court's tendency that the device, to which the IP address was assigned, is identified at the beginning, and it is only later considered who actually had access to the device.

Another high correlation coefficient, with a value of 0.18, can be located between the specification of the device and the stipulation of the IP addresses' time stamp. It is expected that in each case the IP addresses' timestamp should be mentioned, but in the analysis, in only 54% of the cases both the time and date of the IP address' use were listed.

Heat map also shows inverse correlation between expert opinions and final decision. This essentially means that expert opinions used in the court's reasoning usually result in acquittal in criminal proceedings or in action dismissal in civil proceedings. Different types of evidence, like real evidence, documentary evidence, testimony of a witness and testimony of a defendant, and person's relation to the device have similar, although less significant effects. We can also get to this conclusion if we follow in dubio pro reo principle ([when] in doubt, for the accused), according to which the court has to present factual circumstances of the case in favour of the accused in case of doubts. One could therefore assume that the conviction in criminal proceedings or a successful action in civil proceedings and the evidence produced are directly dependent.

The authors have also decided to study this aspect empirically and designed a new attribute - counting the number of types of evidence considered (from 0 to 4). The correlation coefficient of this variable with the attribute guilty showed to be -0.49, which indicates a strong negative relationship. This means that the more different types of evidence are used, the higher the possibility that the defendant will be acquitted or that the civil action will be unsuccessful.

In this part of the analysis, the authors have only considered the decisions which contained a reasoning part. Since the Slovak law allows the reasoning part to be omitted in certain cases, many decisions did not contain it at all. This happens, for example, in cases where the defendant admits his or her guilt in which the court further inspects only the evidence necessary to decide on the type and the length of punishment, but not also the evidence regarding the guilt of the defendant, which is also known as an agreement on crime and punishment. So-called criminal orders also do not contain the reasoning part. In this case, if the legal case is sufficiently strong, the court may opt to give their decision without the case going to trial. From all of the decisions examined, 42 were in the form of an agreement on crime and punishment, another 42 decisions were in the form of a

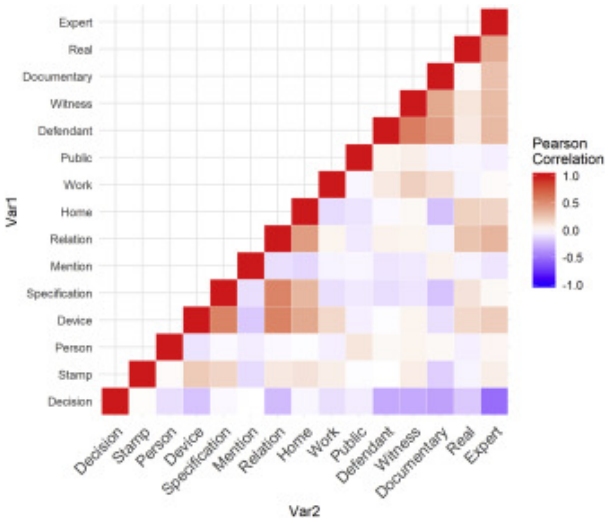


Figure 1: Heat map of correlation coefficient for all pairs of attributes [20]

criminal order and in 26 decisions both the prosecutor and the defendant waived their right to appeal the court's decision. In all of these cases the court only issues a simplified version of the judgement without the reasoning part.

To analyse the decisions, in which the reasoning part was present, the clustering method was used. The decisions examined were classified into four different clusters, as presented in the table 1.

Cluster	Cl. 1	Cl. 2	Cl. 3	Cl. 4
Number of decisions	56	47	28	56
Final decision	98%	89%	46%	96%
Timestamp	71%	87%	54%	9%
Assignment to person	4%	4%	4%	13%
Assignment to device	95%	0%	96%	0%
Spec. of device	54%	0%	21%	0%
Only mentioned	0%	0%	0%	14%
Relationship	39%	0%	50%	0%
Location at home	55%	21%	32%	0%
Location at work	4%	0%	18%	0%
Location in public	2%	2%	0%	4%
Testimony of a defendant	0%	30%	36%	0%
Testimony of a witness	0%	19%	39%	0%
Documentary evidence	0%	43%	93%	41%
Real evidence	4%	2%	14%	0%
Expert opinions	2%	11%	57%	0%

Table 1: Table of clusters [20]

In the first cluster are decisions with a high probability of the IP addresses' timestamps being specified (71%), the IP address being assigned to a device (95%) and the defendant's relationship to the device being presented. In most cases, the device was found in the defendant's home. In these cases, the probability of the court considering other evidence is low (4% real evidence, 2% expert opinion).

In the second cluster of decisions, probability of including the IP addresses' timestamps in the decisions is again high (87%), while the IP addresses are mostly not assigned to a person or a device. The IP addresses were mostly leading to the defendant's home or less often to public places. In contrast to the previous cluster, here it is highly likely that different types of evidence are used, like documentary evidence (43%), defendant's testimony (30%), witnesses' testimony (19%) and expert opinions (11%).

In the third cluster of decisions, the IP addresses used are almost always assigned to a device (96%) and it is relatively common that the relationship between the defendant and the device is regarded (50%) and that the IP address' timestamps are present, while the specification of the device, however, is not. The IP addresses were mostly leading to the defendant's home or place of work. The use of a combination of different types of evidence, specifically documentary evidence (93%), expert opinions (57%), witnesses' testimony (39%), defendant's testimony (36%) as well as real evidence (14%) is also common in this cluster.

In the last cluster of decisions, the IP addresses are only rarely used in the reasoning part (14%), without exploring additional relevant attributes in more detail. The IP addresses are with the highest probability (13%) assigned to a person, while the additional information is rarely provided. In these cases, only the documentary evidence (41%) is likely to be additionally considered.

If an IP address is assigned to a device, the relationship between the device and the person is more likely to be defined, which confirms the authors' theory that the court is firstly trying to link the IP addresses to the corresponding devices and only later to link these devices to a specific person [20].

5.2 Privacy and personal data protection issues

The authors also analysed how the Slovakian courts decided to approach personal data anonymisation in their judgements. Before court decisions become publicly available, all sensitive data must be anonymised. The process consists of either excluding sensitive data or replacing them with initials, random letters, ellipses etc. The authors focused on court decisions that contained the term 'IP address' and classified personal data that they contained into three categories: standard personal data, other online identifiers and IP addresses.

As standard personal data, the authors considered the data that allows direct identification of an individual without requiring any additional information. Such data are first and last name, date of birth, personal identification numbers etc. The authors discovered that such data is anonymised in almost all cases, as the defendant's name could only be found in 6 court decisions, most likely due to a mistake made by an individual responsible for data anonymisation, as this only happened in multiple pages long decisions and in their final parts.

A person can also be identified indirectly by using online identifiers in combination with other data. The authors were researching what kind of online identifiers, apart from an IP address, were used in different court decisions and

whether they were anonymised or not. The most common online identifier was username for signing into social media websites, discussion forums and other websites, which was present in 69 court decisions. The second most commonly used identifier was email address, which appeared in 26 decisions. Other identifiers, such as user ID number, were less common an appeared only in individual decisions. Out of 187 court decisions that were analysed, 112 contained online identifiers, apart from an IP address. In 83 decisions, all online identifiers were anonymised. In 5 cases, in which the decisions contained more than one online identifier, the court decided to anonymise just some online identifiers. In most cases, an email address was anonymised, while a username was not, in authors' opinion probably because the usernames in these decisions consisted of random letters, numbers or names and surnames commonly used in Slovakia and were therefore deemed unable to be connected to a natural person by the court. There were also 25 cases, in which none of the online identifiers were anonymised.

Finally, the authors analysed in how many court decisions the numeric representations of IP addresses were anonymised. In 113 decisions, all IP addresses present were anonymised, in 10 decisions, the numeric representations of the IP addresses were not present and in 6 decisions, which all contained multiple IP addresses, some of them were anonymised while the others were not. As the courts did not give their explanation about this decision and as the authors were unable to find any correlations between which IP addresses were anonymised and which were not, the authors concluded that this was likely a consequence of a mistake made during an anonymisation process. There were also 57 decisions, in which none of the IP addresses present was anonymised [20].

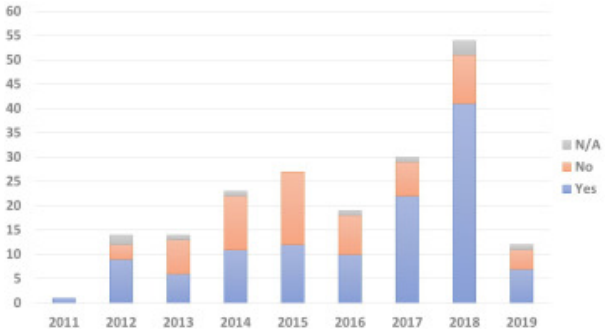


Figure 2: Anonymisation of the IP addresses in court decisions [20]

5.3 Discussion of the related issues

In criminal investigation, the authorities have to be careful not to exceed the country's jurisdiction. The authors decided to examine the localisation of the IP addresses that were not anonymised, which happened in 126 decisions. The majority of these were located within the territory of the Slovak Republic. As the court is not required to change the jurisdiction just on the basis that an IP address is located in a different country, no jurisdiction issues were considered in any of the decisions examined.

The authors also decided to analyse whether the courts differentiate between static and dynamic, public and private and IPv4 and IPv6 IP addresses. In most decisions, the information about whether the IP addresses were static or dynamic, were not present. In 7 decisions the IP address was stated to be dynamic and in 1 to be static. As for public and private IP addresses, it was only mentioned in one decision that an IP address is public, while there was no specific classification in other cases. Finally, the details about whether the IP addresses were IPv4 or IPv6 were not present in any of the examined court decisions, in authors' opinion probably because all IP addresses that were not anonymised were in the form of an IPv4 address [20].

5.4 User identification using IP addresses in Slovenia

Finally, we have examined some of the court decisions related to lawfulness of user identification using IP addresses in Slovenia. We have focused on recent court decisions by the Supreme Court of the Republic of Slovenia and the Higher Courts. Most of the court decisions that we have examined were answering the question if the information about the owner of an IP address was legally obtained by the authorities. Additionally, almost all of these also discussed the difference between static and dynamic IP addresses.

In 2006, the Swiss police discovered child pornography among the users of a peer-to-peer network and identified a dynamic Slovenian IP address among the IP addresses of the users of said network. They passed it to the Slovenian authorities who then obtained the information about the subscriber from the internet service provider. At the address of a subscriber, the police seized four computers in a house search, found pornographic content on the hard drives and based on this and the statements of the family members on how the computers were used the suspicion shifted from the subscriber to his son, who was later also convicted of owning and sharing pornographic content. The accused, however, filed an appeal to the Supreme Court of the Republic of Slovenia, stating that the authorities should have obtained a court order to get the information about the owner of the IP address. This data was considered traffic data at the time, for which the authorities should have obtained a court order prior to getting them. The authorities defended themselves by stating that they had only requested information about a user of a specific IP address at a specific time, which by itself was not a traffic data, thus not requiring a court order to obtain them, but only a written request from the police. The court decided that the authorities should have obtained a court order prior to requesting the data about the user of the IP address, as the defendant never decided to publicly share it. The conviction was therefore invalidated, the illegally obtained data excluded from the evidence and the case remanded [5].

A similar case happened in 2010, when Interpol Luxembourg investigated a server on which they discovered child pornography. The Luxembourg police found a Slovenian IP address among the ones that were used to access the illegal content and passed it to Slovenian police together with the date and time of when the access to the illegal content occurred. The Slovenian authorities requested the information about the owner of the IP address at that time. The internet service provider passed the information about the owner of the

static IP address to the authorities, which was then used to identify the suspect. In a house search at the home of the suspect, the police seized the hard drives. In the analysis they discovered that the drives contained child pornography. The suspect was convicted by the District Court Maribor, but filed an appeal, stating that the authorities should have obtained a court order to get the data about the owner of an IP address. In 2018, the Maribor Higher Court decided in his favour, stating that the European Convention on Human Rights was violated and that the authorities should have obtained a court order prior to obtaining the information about the owner of the IP address [3].

A year after this case in two different procedures, the defendants also claimed that the authorities should have obtained a court order prior to collecting the information about the owners of the IP addresses. The IP address was identified as static in both cases. In both cases, however, Ljubljana Higher Court decided that a court order was not required to collect this information, stating that the defendants had been using a static IP address, which does not require browsing through internet traffic data to link an IP address to an individual because the internet service provider already has a record of static IP addresses and their owners. Although the court's decision in the previously described case was even mentioned in one of the judgements, it was also stated that it was not binding [1, 2].

In 2013, a defendant was accused of verbal abuse by posting a comment on a news media website. She was identified via an IP address which had been saved by a website. The defendant claimed that the news media website should not had presented her IP address as an evidence and stated that by this her right to privacy according to the Constitution of Slovenia had been violated. The Supreme Court of the Republic of Slovenia refused her appeal, deciding that the defendant waived her right to privacy by posting a comment on a publicly available website. The court did not unambiguously specify, though, if the defendant was using a dynamic or a static IP address [4].

Based on the examined recent juridical practice in Slovenia, we can say that the steps that should be taken when identifying the individual using an IP address substantially depend on whether the IP address in question is static or dynamic. The courts generally agree that the users of dynamic IP addresses can expect a higher degree of privacy by default, as their IP address is changing through time and unlike static IP address, it is not permanently bound to a subscriber. In most court decisions that we have examined, it is stated that a court order is required to request the information about the owner of a dynamic IP address. However, the courts are more divided in cases where the defendants were using static IP addresses, as in some cases the courts decided that a court order is necessary also for obtaining the information about the owner of a static IP address, while in some cases the courts decided that a written request from the police is enough.

This grey area regarding how should the information about the owners of static IP addresses be obtained from the legal point of view poses problems for the authorities as they can not be certain whether or not they need a court order to obtain this information. The issue is extremely important, as illegal acquisition of the evidence has substantial consequences. In addition to the exclusion of the illegally obtained evidence, all evidence that was obtained as a result of an illegally obtained evidence has to be excluded as well, under the fruit of the poisonous tree doctrine.

It is also important to mention that the IP address by itself can not be used to directly accuse the subscriber, as multiple devices and, consequently, individuals may be using it. The information that may be obtained from it, however, can be used to point to a suspect or to obtain additional proof for analysis, usually computers or other devices, as seen in the first two cases described. In the last case, though, it was implied that the accused was identified as a suspect only by her being the subscriber to whom the IP address was assigned, which could be problematic at the trial.

6. CONCLUSION

The authors' research resulted in very interesting discoveries. One of the more unusual ones was that the more evidence is produced, the higher the possibility that the defendant is cleared or the civil action unsuccessful. It is expected that the court needs more evidence to determine the defendant's guilt and not innocence, in respect to in dubio pro reo principle.

Another finding of the research is, that court decisions still often connect the IP addresses to persons, although the IP addresses primarily identify the devices, therefore the authors suggest more emphasis on the formation of a connection between an IP address and a natural person through the device with the IP address in question. Another problem is the absence of the device specifications and the deliberations on whether any other persons might have had access to the device beside the defendant from court decisions. The authors believe it would be appropriate to reach some sort of standardisation of the use of IP addresses as evidence in court proceedings.

As for the anonymisation of IP addresses, the authors found that this poses a challenge for the courts, as their approaches to this issue vary. One of the possible solutions to this problem could be the adoption of recommendations by Ministry of Justice.

Our own research of some recent court decisions in Slovenia showed that the courts do not unanimously agree on whether a court order is necessary for authorities to obtain the information about the owner of an IP address. The most interesting discovery were three court cases in a time frame of one year, each one of them discussing the question whether a court order should is required to obtain the information about the owner of a static IP address, in one of which the court decided that a court order was necessary to obtain this information and in the other two the court decided otherwise. To avoid this issue in the future, we suggest the Ministry of Justice to establish the guidelines on this matter. As IP addresses can not be unambiguously linked to a single person, we also suggest that the IP address is only used to identify the subscriber and that additional evidence is collected and analysed before the accusation.

7. REFERENCES

[1] VSL Sklep V Kp 1896/2017. http://sodnapraksa.si/?q=ip%20naslov&database[IESP]=IESP&_submit=i%C5%A1%C4%8Di&rowsPerPage=20&page=0&id=2015081111435304. [Online, accessed: 7 May, 2021].

[2] VSL Sodba III Kp 16465/2017. http://sodnapraksa.si/?q=ip%20naslov&database[IESP]=IESP&_submit=i%C5%A1%C4%8Di&rowsPerPage=20&page=0&id=2015081111428940. [Online, accessed: 7 May, 2021].

[3] VSM Sklep II Kp 50396/2011. http://sodnapraksa.si/?q=ip%20naslov&database[IESP]=IESP&_submit=i%C5%A1%C4%8Di&rowsPerPage=20&page=0&id=2015081111423267. [Online, accessed: 7 May, 2021].

[4] VSRS Sodba I Ips 27119/2014. http://sodnapraksa.si/?q=ip%20naslov&database[SOVS]=SOVS&_submit=i%C5%A1%C4%8Di&rowsPerPage=20&page=0&id=2015081111402694. [Online, accessed: 7 May, 2021].

[5] VSRS Sodba I Ips 31751/2018. http://sodnapraksa.si/?q=ip%20naslov&database[SOVS]=SOVS&_submit=i%C5%A1%C4%8Di&rowsPerPage=20&page=0&id=2015081111438110. [Online, accessed: 7 May, 2021].

[6] R. Braden et al. Requirements for internet hosts–communication layers. *STD 3, RFC 1122, October*, 1989.

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

[8] K. A. Cole, S. Gupta, D. Gurugubelli, and M. K. Rogers. A Review of Recent Case Law Related to Digital Forensics: The Current Issues. *Annual ADFSL Conference on Digital Forensics, Security and Law*, 2015.

[9] E. Committee. E.C. On Legal Co-Operation. *European committee on legal co-operation.*, 2016.

[10] T. Gřivna and J. Drápal. Attacks on the confidentiality, integrity and availability of data and computer systems in the criminal case law of the Czech Republic. *Digital Investigation*, 28:1–13, 2019.

[11] Hoschar v. Layne, 647 F. App'x 632, 632, 2016.

[12] Katz v. United States, 389 U.S. 347, 361, 1967.

[13] M. Keyser. The Council of Europe Convention on Cybercrime. *J. Transnat'l L. & Pol'y*, 12:287, 2002.

[14] J. M. Kizza, Kizza, and Wheeler. *Guide to computer network security*. Springer, 2013.

[15] E. Larson. Tracking Criminals with Internet Protocol Addresses: Is Law Enforcement Correctly Identifying Perpetrators? *North Carolina Journal of Law & Technology*, 18(5):316, 2017.

[16] P. Lundevall-Unger and T. Tranvik. IP addresses–Just a number? *International Journal of Law and Information Technology*, 19(1):53–73, 2011.

[17] S. Menard. *Applied logistic regression analysis*, volume 106. Sage, 2002.

[18] R. Montasari. Digital Evidence: Disclosure and Admissibility in the United Kingdom Jurisdiction. In H. Jahankhani, A. Carlile, D. Emm, A. Hosseinian-Far, G. Brown, G. Sexton, and A. Jamal, editors, *Global Security, Safety and Sustainability - The Security Challenges of the Connected World*, pages 42–52, Cham, 2016. Springer International Publishing.

[19] L. Rokach and O. Maimon. Clustering methods. In *Data mining and knowledge discovery handbook*, pages 321–352. Springer, 2005.

[20] P. Sokol, L. Rózenfeldová, K. Lučivjanská, and J. Harašta. IP addresses in the context of digital evidence in the criminal and civil case law of the Slovak Republic. *Forensic Science International: Digital Investigation*, 32:300918, 2020.

[21] United States v. Johnson, No. 15-00340-01-CR-W-GAF, 2016.

[22] United States v. Michaud, No. CR15-5351RJB, WL 337263, 2016.

[23] M. von Grafenstein. *The Principle of Purpose Limitation in Data Protection Laws*. Nomos Verlagsgesellschaft MbH, 2018.

DeepUAge: Improving Underage Age Estimation Accuracy to Aid CSEM Investigation

Kastriot Kadriu
Faculty of Computer and Information Science
University of Ljubljana
Ljubljana, Slovenia
kk5222@student.uni-lj.si

Blin Beqiri
Faculty of Computer and Information Science
University of Ljubljana
Ljubljana, Slovenia
bb5756@student.uni-lj.si

Milenko Obradovic
Faculty of Computer and Information Science
University of Ljubljana
Ljubljana, Slovenia
mo5325@student.uni-lj.si

ABSTRACT
Age estimation models are useful for identifying possible materials involved in Child Sexual Exploitation Material (CSEM) distribution. These models are build using deep learning networks. They are trained over specific datasets. Working with datasets, especially for such cases, is a challenge on its own due to the fact that much of the data isn't labelled or is mislabeled. Although the accuracy of similar models has been improved over the years, there are still challenges they face, some of which will be mentioned in this paper. In this paper, we go back to the basis of building such model, highlight the features and jump onto challenges it faces.

Keywords
deep learning, face recognition, age estimation, digital forensics

1. INTRODUCTION
Age as a soft biometric trait is difficult to predict due to discrepancies between face and body features, absence of reliable cues, natural variation regarding variability across different ethnicities, and the environment where the victim appears. The increasing accuracy of deep learning models, have made them useful in self identifying tasks in real life. The way those models are trained is based on some pre-labeled data, which based on some probabilistic model depict what label the test data will have. Due to the unpredictable nature of data, amongst the issues, this labeling isn't always right.

2. THE MODEL
There are different ways to build deep learning models, but ones that have been proven to achieve high accuracies in the field of image recognition are the Convolutional Neural Networks (CNN). These models are based on the mathematical operation of convoluting. A basic architecture of a CNN model would consist of different layers in which the data will go through and get transformed, those layers are convolutional, pooling and a dense layer.

2.1 The dataset
When working with a machine learning model, dataset is one of, if not the most, crucial component(s). Data mining is a whole other step of the process, but sometimes, if the area of application isn't too narrow, we might count on some public dataset for use in our models.
The columns of the dataset are referred to as features. In a classification problem, we have to map a set of features f, which represent a row, with a class (label) c. In a regression problem, we have to map a set of features f, with a continuous output variable.
Any of these problem types can be adjusted for our needs, however the processing of features and results is different for both of them.
The dataset is used for training a model, that is generating probabilistic data, that could later be used for making predictions based on some testing data. Testing data could be some custom and external data or we can split the working dataset into a train-test split that could be e.g in a 80/20 ratio.

2.2 The architecture
2.2.1 Convolution layer
A convolutional layer applies a filter that results in an activation. This filter, also known as kernel, is applied multiple times in an input returning a feature map. A convolutional layer can have multiple filters. Filter size is not arbitrary but it has to be smaller than the input size. When configuring the layer, the stride parameter needs to be set as well. Stride is the number of columns the filter moves for. If the stride is 2, that means that the filter is sliding for two columns right. An alternative approach to applying a filter to an input is to give each number in the matrix the chance to be in the center of the filter. That is set up by the padding parameter. If chosen, this parameter adds zeros around the input. When the features have been mapped, they go through a nonlinearity function. Such function is the rectified linear activation function - ReLU. ReLU is a linear function that outputs the input unless it's negative in which case it outputs zero.

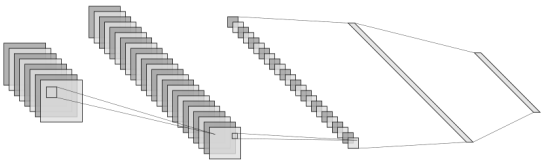


Figure 1: CNN model architecture

2.2.2 *Pooling layer*

A pooling layer reduces the dimensionality of a feature map by summarizing those inputs to a smaller dimension. The two most common pooling types are: Max and Average pooling.

Average pooling calculates the average value for each patch on the feature map. This average value represents the features in a smooth way, which may not be very useful for feature detection.

Max pooling takes the maximum value from each patch. This is useful because it is able to extract the most important features such as edges. When dealing with image recognition, this is the most used.

Other parameters, although not exclusive to CNN models, are **epoch** and **batch-size**. Epoch is the number of iterations we will train our model for, whereas batch-size is the number of samples that will be propagated through the network.

2.3 Challenges

When dealing with such models a number of challenges can arise.

- **Imbalanced dataset** - Most of these challenges are caused by the dataset, which can be imbalanced, leading to a biased model e.g if we train our age estimator with a dataset where most of images are "old" people then the model will be biased to predict an "old" age for subsequent test images.
- **Noisy data** - Dataset can also be noisy, e.g our images contain things other than a face, which is what we should be interested on, thus "confusing" our model. Those challenges become even larger in our case because images are harder to pre-process, so we can not count much in this phase to help us with the issues described above.
- **Lack of data** - The biggest challenge is the lack of data. This lack of data is also related to the privacy issues regarding to the type of data we are working with because, unlike text where you can attach some anonymity to it, in images this anonymity which would take care of our privacy concerns would be harder to be achieved, and especially in our case where we are dealing with real underage human faces, which sparks ethical implications.
- **Computational limit** - Images are space consuming data, and building a model from a dataset that could contain hundreds of thousands or even millions of records, requires computational power and storage that might not be available easily.

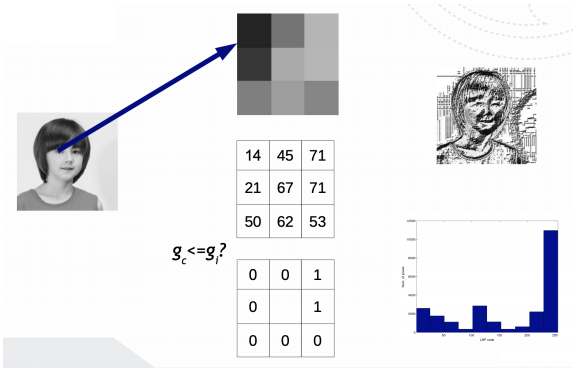


Figure 2: A sample of how image is treated and manipulated in a CNN model [3].

3. DEEPUAGE MODEL

3.1 Configuration

In this model, the problem is regarded as a classification one rather than a regression one, which means the model aims to classify an image to an exact age rather than an age range. The model has over 50 layers. The last Softmax layer with 1000 outputs has been replaced by a soft-max activation layer of 20 outputs that represent the final age classes. A custom dataset of around 17000 records has been built from different sources where the data was labelled in age and gender features. This dataset is compromised of the VisAGe [6], FGNET [7], IMDB-WIKI [8], FERET [9] and MEDS [10] data sources. This model was built accordingly with balanced classes.

3.2 Model

Facial images were pre-processed to filter unneccessary features that could potentially affect the learning process of the model. Rather than removing those unneeded features, the needed features were extracted and underwent some further calculation. The idea was to extract the features, also referred to as landmarks, calculate them and train the classification model over these calculations.

A three step preprocessing procedure was implemented: face detection, facial landmark collection and facial normalisation. Face detection was done using an external toolkit[1]. Facial landmark collection represents the main focus of the pre-processing work. The model tried different approaches for collecting the landmarks. Those approaches were the **Face++** service and the **dlib** machine learning library. Face++ has a higher performance collecting the landmarks however the issue with it is that it's a remote cloud service, and the transmission of our training data which is thought to be a sensitive information, would pose a new obstacle in the architecture and would be somewhat of a privacy risk.

Using dlib library, it was possible to derive a new approach for pre-processing the data - the DCA approach. The strengths

[1]Face detection toolkit developed by VIPL lab of CAS

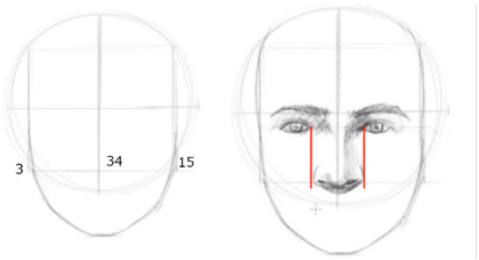


Figure 3: Facial landmark reconstruction using the Loomis face proportion approach [1].

of this approach include the fact that it is able to account for forehead or wrinkles in its landmarks detector - features that impact the age estimation of the subject. After collecting the landmarks, the forehead and hairline's points (which aren't accounted in other methodologies) have been calculated in a process that emulates the face drawing methodology. Those steps are as follows:

- Obtain coordinates x, y of the lowest point of the nose
- Compute the average distance between x, y and the intersecting points l_{left}, l_{right} of the line that is perpendicularly drawn from the nose pointing towards the countour with the following formulas

$$d_1 = \sqrt{(l_{left_x} - x)^2 + (l_{left_y} - y)^2}$$

$$d_2 = \sqrt{(l_{right_x} - x)^2 + (l_{right_y} - y)^2}$$

  Then the square sides surface is calculated as:

$$square_{side} = \frac{d_1 + d_2}{2} * 2$$

- Using the gained values, the landmarks are re-constructed using the foundations of the Loomis face proportion approach (Fig 3).

Those pre-processing procedures were also done with the intention of minimizing the negative impact of a possible noise in the training pictures, for example, for cases where the picture isn't a straightforward profile picture.

3.3 Performance

The model was evaluated using the mean average error (MAE). On a 80/20 split for training and testing data respectively, it achieved a MAE value of 2.73.
DeepUAge model ranks in the top 3 performers of the models in the same fields, according to their MAE values, the other two being Microsoft Azure API, followed by Amazon Recognition (Fig 1).
Despite performing high, the logarithmic loss of the model (which shows the variation of the actual label from the machine learning model predicted value) of 1.79. The aim is to reduce it to 0.

A limitation of this model is that it hasn't been trained on diverse data, that is a demographic disbalance in regards to ethnicity. In addition, it would be better if the dataset contained more detailed features other than age and gender.

Technique	Result (MAE)
DeepUAge	**2.73**
Azure Face	3.60
Amazon Rekognition	3.74

Table 1: Evaluation results using the MAE metric [1].

4. RELATED WORK

Microsoft Azure Face API [11] provides an algorithm that is able to detect, recognize and analyze human faces in images. The API is able to verify e.g are these two images the same person, to identify e.g is the face from this image present in a database record, find similar faces and generate groups for them. These powerful feature are privacy concerning however Microsoft has announced that it will not sell its facial recognition technology to police departments in the United States (the jurisdiction where the company is based in) until strong regulations have been defined. The service is available for integration as a paid service.
Another paid service is Amazon Rekognition [12]. Amazon Rekognition has similar features like Azure Face, but in addition it also offers supports for custom labeling, celebrity detection and personal protective equipment (PPE) detection, the latter including a use-case where people wear masks that is relevant to today's situation. Amazon Rekognition uses DetectFaces API to detect faces. Up to 100 faces can be detected from an image with this API, and in addition it returns a confidence value that the bounding box contains a face, and facial landmarks. An advantage of this service is that it's supported by a greater ecosystem that is able to integrate the service in a more useful way via other services that Amazon offers.
Face++ [13] is another important service in the field, that was one of the approaches that the DeepUAge model considered for the model. It employs the main features of the aforementioned services, but what's unique here is the support for the human body recognition and the ability for 'face merging' from two different sources.
Jung, Soon-Gyo *et al.* in [5], offer an evaluation of the accuracies of the mentioned services for face, gender and age detection. The models have been tested in some benchmark datasets: 100 Celebrities Dataset, IMDb-Wiki Dataset, Twitter Profile Photo Dataset. The age detection has been done in *bins* of age ranges from 13 to 65 year old. From the results in table 2, we can see that models generally struggle with proper age detection and that makes the importance of DeepUAge even greater.
Other than those services, there has been scientific research with the intention of coming up with a sustainable way for recognizing age. Boys2men [3] describes a model built with the intention of automatic detection of enfants in pornography content. The model extracts features for each picture as local descriptors and then manipulates them accordingly. The model is challenged by a missing of a *sustainable* dataset, but it does lay ground for a logic that would gain

Dataset	Face++	Amazon	Azure Face
100 celebrities	0.28	0.35	0.37
IDMb-Wiki	0.34	0.3	0.45
Twitter	0.41	0.36	0.66
All datasets	0.34	0.3	0.45

Table 2: Age detection evaluation results using a simple accuracy metric [5].

great use in the digital investigation field.
Ramanathan and Chellappa (2006) proposes a model that identifies age progression based on the face shape. The model is able to account for gender based differences. Using anthropometric data, the model is also able to characterize facial growth on people from different ethnicities. The downside of this proposed model, is that it fails to include textural *layers* across age. For example, facial hair or the fat tissue change isn't accounted for, those characteristics being very common amongst teenagers.

5. DIGITAL FORENSICS PERSPECTIVE

Law enforcement is a trade off between the cost and results, known as a 'balance scorecard' [15]. Hiring better specialists improves the quality of investigations, but increases the budget needed to mantain law enforcement agencies, which could result being in a crushing expense for smaller police departments.
Considering the increasing needs for digital forensics analysis, paired with a lack of people with expertise in many law enforcement agencies, the need for such models is imminent. The use of automated models is a subject to, not only technological, but also political and social discussions. Automated digital forensics sometimes is referred to as push-button forenscics (PBF). Criticism of PBF revolves around the over reliance on it while leaving little to no room for expert's opinions. Automated analysis of evidence might happen in a faster pace that a human contributor might not be able to follow. And obviously, people working in this field are afraid automation will take over their jobs and this brings us to a general issue about the use of AI in our lives. Despite the skepticism, automation is already integrated in the digital forensics procedures to some extent. There are multiple tools that help the work of the investigator such as Autopsy [16] and the Forensic Tool Kit [17]. At this point, the automated tools are used for finding and converting data to information that is then used by the investigator to manually make some conclusion.
An advantage of automation would be that it would speed up the investigation process, minimizing any opportunity for the bias or prejudice of people involved in the investigator to influence the process.
Child exploitation investigations are one of the most common investigation types in digital forensics laboratories. The usage of those automated models not only would increase the effectiveness of those investigations, which are already hindered by an increased use of anonymization tools, but as an automated process, would reduce the work exposure to incriminating archives of indecent images, therefore, reducing the psychological ramifications.
Going back to the current state, even if we ignore the objections from different sides, the automation world has more work to do before it can be integrated fully in an investigation. It needs to be able to interpret what information means as well and not just find it like it currently does.
Regardless of that, the promising results of DeepUAge make it a potential aid for law enforcement. If not actively used for investigation, it could at least be used as an evidence analysis priorisation tool.

6. CONCLUSION

In this paper, we were introduced to deep learning models for classification and the challenges they face.
DeepUAge model achieves state-of-the-art performance and is able to compete with a few other models in the field. The strengths of this model is the feature's computation. It includes a new approach for constructing the facial landmarks needed for age detection. On the other hand, the model is hindered by a lack of a larger and more detailed dataset.
In a world of big data, the data might become more easily available in the future, but at a computational cost as the storage requirements are going to be increased.
As we saw from the Related Work section, some big companies were able to afford the infrastructure for creating resource consuming models, but even then, their accuracies for age detection were low. Relevant data acquisition was a struggle that has potential to be solved in the future, but on the other hand, the lack of a sustainable algorithm or logic for age calculation might never really be achieved as humans (and their faces) are unpredictable.
Digital forensics field could use help from several automated methods. That could reduce the issues that are caused by the lack of human resources and on the other hand, could reduce the psychological load the work of a digital forensics investigator has. The future work regarding the integration of automated tools in the digital forensics investigations needs to be concerned about upgrading the accuracy of those tools and the establishment of relevant regulations for their use that seem to be missing in most jurisdictions.

7. REFERENCES

[1] Anda, Felix. Le-Khac, Nhien-An. Scanlon, Mark. *DeepUAge: Improving Underage Age Estimation Accuracy to Aid CSEM Investigation*

[2] Anda, Felix. Lillis, David. Le-Khac, Nhien-Ah. Scanlon, Mark. *Evaluating Automated Facial Age Estimation Techniques for Digital Forensics*

[3] Castrillon-Santana, M., Lorenzo Navarro, J.J., *Boys2Men, C., 2016. An age estimation dataset with applications to detect infants in pornography content. In: First International Workshop on Biometrics and Image Forensics. IEEE.*

[4] Ramanathan, Narayanan. Chellapp, Rama. *Modeling Age Progression in Young Faces*

[5] Jung, Soon-Gyo *et al. Assessing the Accuracy of Four Popular Face Recognition Tools for Inferring Gender, Age, and Race.*

[6] Anda, F., Lillis, D., Le-Khac, N.A., Scanlon, M. *Evaluating automated facial age estimation techniques for digital forensics.* In: 2018 IEEE Security and Privacy Workshops (SPW). IEEE, pp. 129e139.

[7] Lanitis, Andreas, and Tim Cootes. *Fg-net aging data base.* Cyprus College 2.3 (2002).

[8] Rothe, R., Timofte, R., Van Gool, L., 2018. *Deep*

expectation of real and apparent age from a single image without facial landmarks. Int. J. Comput. Vis. 126 (2), 144e157. https://doi.org/10.1007/s11263-016-0940-3.

[9] Phillips, P.J., Wechsler, H., Huang, J., Rauss, P.J., 1998. *The FERET database and evaluation procedure for face-recognition algorithms.* Image Vis Comput. 16 (5), 295e306.

[10] Founds, A.P., Orlans, N., Genevieve, W., Watson, C.I. *NIST Special Database 32-Multiple Encounter Dataset II (MEDS-II).* Tech. Rep. NIST. (2011)

[11] *Azure Face API.* Retrieved from https://azure.microsoft.com/en-us/services/cognitive-services/face/ on May 2021.

[12] *Amazong Rekognition.* Retrieved from https://aws.amazon.com/rekognition/resources/ on May 2021.

[13] *Face++.* Retrieved from https://www.faceplusplus.com/ on May 2021.

[14] James I., Joshua. Gladyshev, Pavel. *Challenges with Automation in Digital Forensic Investigations.*

[15] Ashworth, J. *Is there a link between quality standards & performance improvement?, Home Office.* (2010).

[16] *Autopsy* Retrieved from https://www.sleuthkit.org/autopsy/ on May 2021.

[17] *Forensics toolkit* Retrieved from https://accessdata.com/products-services/forensic-toolkit-ftk on May 2021.

Prepoznavanje globokih ponaredkov z odkrivanjem obraznih artefaktov

Bor Brecelj
Fakulteta za Matematiko in Fiziko
Jadranska ulica 19
1000, Ljubljana
bb3263@student.uni-lj.si

Deni Cerovac
Fakulteta za Računalništvo in Informatiko
Večna pot 113
1000, Ljubljana
dc9981@student.uni-lj.si

Roni Likar
Fakulteta za Računalništvo in Informatiko
Večna pot 113
1000, Ljubljana
rl8622@student.uni-lj.si

POVZETEK

V seminarski nalogi opisujemo odkrivanje globokih ponaredkov (angl. deepfake) na podlagi odkrivanja artefaktov, ki nastanejo pri ponarejanju posnetka. Zato bomo najprej naredili pregled področja in ugotovili, kakšne tehnike se danes uporabljajo za odkrivanje globokih ponaredkov. Nato bomo opisali, kaj je globok ponaredek, ter kakšne vrste artefaktov lahko nastanejo pri delanju globokih ponaredkov. Na koncu pa si bomo pogledali implementacijo modela za zaznavanje globokih ponaredkov na podlagi odkrivanja artefaktov, ki je podrobneje opisana v članku [5].

Ključne besede

globoki ponaredki, artefakti, detekcija, strojno učenje

1. UVOD

Z napredki umetne inteligence in računalniške grafike so seveda napredovali tudi ponaredki posnetkov oziroma kasneje tako imenovani globoki ponaredki, ki z uporabo globokih nevronskih mrež ustvarijo posnetke ene osebe z obrazom druge. Z napredovanjem in izboljšavo posnetkov je splošna javnost postala zaskrbljena in začeli so se spraševati, ali lahko sploh še zaupajo medijski vsebini na internetu. Za zdaj tehnologija globokih ponaredkov še ni toliko uspešna, da se je ne bi dalo prepoznati. Za prepoznavo imamo razne artefakte in druge napake, ki to izdajo. Te napake je mogoče še zaznati brez pomoči raznih orodij v primerih, ko globoki ponaredki niso najbolj kakovostni. Ko pa bo tehnologija napredovala, je verjetnost prepoznave globokih posnetkov brez orodij zelo majhna.

2. PREGLED PODROČJA

V prvemu izmed prebranih člankov lahko najdemo raziskavo in opis novih metod z globokim učenjem, ki efektivno prepoznajo lažne video posnetke generirane z umetno inteligenco [5]. Zasledili smo tudi pregled algoritmov za sledenje in spremembo obrazov, ki izpostavijo artefakte, kateri spominjajo na računalniške probleme sledenja in spreminjanja obrazov [6]. Eden izmed zanimivih pristopov je tudi formulacija prepoznavanja globoko ponarejenih posnetkov kot drobnozrnat klasifikacijski problem in predlog nove več-pozornostne mreže za prepoznavo globokih ponaredkov [10]. Zelo zanimiva je tudi tehnika identifikacije majhnih sprememb posnetkov obraza, ki so posledica sprememb krvnega toka [2]. Zelo uporaben je bil tudi članek, ki predstavi metode globokega učenja za prepoznavo računalniško ustvarjenih grafičnih slik in posnetkov od pravih z uporabo konvolucijskih nevronskih mrež [7]. Med drugim je zelo učinkovit pristop tudi prepoznavanje umetno generiranih posnetkov s pomočjo nevronskih mrež glede na mežikanje osebe na posnetku [4].

3. GLOBOKI PONAREDKI

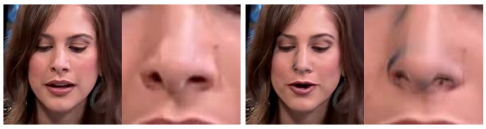
Skozi pretekla leta je področje globokih ponaredkov doživelo velik razcvet. Socialna omrežja so preplavili zelo prepričljivi globoki ponaredki kot na primer TikTokov uporabnik @deeptomcruise (slika 1).

Slika 1: Zelo prepričljiv globok ponaredek Toma Cruisa

Prav zaradi takih posnetkov, kjer povprečen uporabnik omrežij ni mogel preveriti, ali je na posnetku prava oseba ali ne, so se začela porajati vprašanja, kako sploh še zaupati slikam ali posnetkom na internetu. Ob veliki večini takih posnetkov je njihov glavni namen ustvariti lažne dokaze znanih oseb in s tem vplivati na javno mnenje o vpletenih osebah na video posnetkih.

Ustvarjanje globokih ponaredkov lahko v splošnem zelo dobro uspe, ampak postavitev oseb v določen kontekst zelo oteži postopek ustvarjanja posnetkov. Takšne težave pa forenzikom zelo olajšajo raziskovanje in dokazovanje verodostojnosti posnetkov. Napake na posnetkih so vidne v obliki

artefaktov, zaradi katerih lahko ponaredke prepoznamo. Artefakti so grafične napake oz. anomalije, ki se na sliki ne bi smele pojaviti, saj tako deluje oseba nerealistična. Artefakti se v veliki meri pojavljajo okoli najbolj kompleksnih delov obraza, kot so nos, oči, usta, obrvi, zobje, ter na robovih obraza. Pri delanju globokih ponaredkov se lahko pri delih ust pojavljajo bolj zamegljeni deli posnetka. Na nosu se pogosto pojavijo napake nenatančnih oblik in karakteristik nosu. Poleg tega pa se lahko napačno prikazuje tudi odboj svetlobe, ki ni v skladu z ostalimi deli obraza. Dodaten problem artefaktov so tudi zobje, ki se pokažejo kot ena celota namesto skupek manjših delov (slika 2).



Slika 2: Prikaz artefakta na nosu [6]

Eden izmed pomembnejših artefaktov nastane tudi v očeh. Oči so velikega pomena, saj dajo osebi tisto posebno življenjsko podobo. Problem artefaktov se pojavi zaradi odboja svetlobe v očeh. Po kreaciji globoko ponarejenih posnetkov se resolucija pogosto zmanjša, kar posledično tudi zmanjša odsev v očeh. Zaradi zmanjšanja in pogosto tudi odsotnosti odseva v očeh dobijo osebe v posnetkih oči, ki delujejo nerealistične oz. neživljenjske.

Skratka globoko ponarejanje je prilagajanje karakteristik tujih obrazov na obraze oseb v želenih video posnetkih z namenom ustvarjanja lažnih posnetkov [6].

4. ARTEFAKTI

Za prepoznavanje globoko ponarejenih posnetkov si pomagamo z artefakti. To so napake, ki kažejo, da posnetek ali slika ni resnična. Nekatere artefakte lahko opazimo že s prostim očesom, medtem ko orodja za prepoznavanje globokih ponaredkov temeljijo na strojnem učenju. V članku [4] so ugotovili, da je odsotnost mežikanja v posnetkih dober artefakt za njihovo prepoznavanje. Artefakte lahko razdelimo v različne skupine, ki so podrobneje opisane v naslednjih podpoglavjih.

4.1 Napake v osvetlitvi

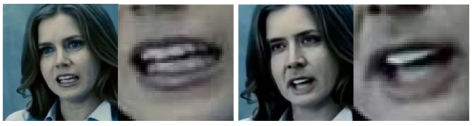
Ta skupina artefaktov zajema odboj svetlobe in sence. Globoko ponarejeni posnetki ponavadi dovolj realistično določijo odboj svetlobe na koži, da jih na podlagi tega ni mogoče ločiti. V nekaterih primerih sence, predvsem okoli nosu, niso dovolj realistične. Najbolj pogost artefakt iz te skupine je odsev v očeh, ki ga ni ali pa ne izpade realistično. Primer je prikazan na sliki 3, kjer se vidi, da je del problema tudi v tem, da je globok ponaredek omejen glede ločljivosti izdelanih posnetkov.

4.2 Geometrija obraza

V to skupino artefaktov sodijo neskladnosti na robu obraza, ki nastanejo zaradi stika med dvema različnima slikama. V novejših globoko ponarejenih posnetkih je to že uspešno rešeno, še vedno pa prihaja do neskladnosti okoli nosu. Najbolj pogost artefakt iz te skupne so zobje, kjer ni modeliran

Slika 3: Prikaz odseva v očeh na resničnih slikah na levi in globoko ponarejeni posnetkih na desni [6]

vsak zob posebej, ampak so modelirani kot celota, kar je prikazano na sliki 4.



Slika 4: Prikaz, kako so modelirani zobje v globoko ponarejenih posnetkih [6]

4.3 Napake pri afinih transformacijah obraza

Globoki ponaredek sestavi novo sliko tako, da iz prve (izvirne) slike vzame le obraz in ga nalepi na drugo (ciljno) sliko. Pri tem mora najti točen položaj obraza na ciljni sliki, da lahko obraz iz izvirne slike natančno prilepi. Za to mora nad obrazom iz izvirne slike narediti afino transformacijo (zasuk, povečava, itd.), zaradi katere lahko pride do razlik v ločljivosti med obrazom in okolico, kar je uporabljeno v metodi, ki jo bomo opisali v naslednjem poglavju.

5. PREPOZNAVANJE GLOBOKIH PONAREDKOV

Tehnologija globokih ponaredkov se je danes tako dobro razvila, da je skoraj nemogoče ločiti med lažnimi in pravimi slikami ter posnetki. Kljub temu pa posnetki vsebujejo določene napake oziroma artefakte, s katerimi lahko ločimo med lažno in pravo sliko. Metode, ki se uporabljajo za odkrivanje artefaktov, so podrobneje opisane v poglavju Artefakti. Ker so napake artefaktov komaj opazne za prosto oko, uporabljamo za odkrivanje le teh strojno učenje. Med obilico strojnih metod so se za tovrstni problem najboljše izkazale konvolucijske nevronske mreže (CNN). Konvolucijske nevronske mreže lahko zaznajo prisotnost artefaktov, ki se pogosto pojavijo v okolici obraza. Zato jih lahko uporabimo kot binarne klasifikatorje, da ugotovimo, ali je slika prava ali ponaredek. V tem poglavju bomo podrobneje opisali model, ki so ga uporabljali za odkrivanje artefaktov v članku [5].

5.1 Nabori podatkov

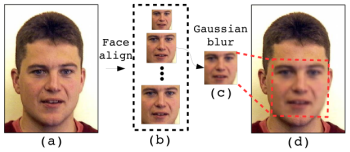
Za učenje ločevanja med originalnimi posnetki in globokimi ponaredki uporabljamo nevronske mreže z binarno klasifikacijo. Za učenje nevronskih mrež uporabimo učne podatke, na katerih nevronsko mrežo naučimo ločevanja med originalom in globokim ponaredkom. Naučene modele nato testiramo na testnih podatkih in preverimo točnost modela.

Ker učimo nevronske mreže ločevanja med originalnimi in globokimi ponaredki, potrebujemo vzorec originalnih slik, imenovan pozitiven vzorec in vzorec globokih ponaredkov, imenovan negativen vzorec.

Za učenje in testiranje modelov so avtorji članka [5] uporabili že pripravljen nabor podatkov. Modele so testirali na posnetkih iz UADFV, VidTIMIT in DeepfakeTIMIT. UADFV vsebuje 98 različnih posnetkov, od tega je 49 resničnih, 49 pa ponarejenih. V vsakem posnetku je ena oseba in vsi posnetki trajajo približno 11 sekund. Drugi nabor podatkov, ki se uporablja za strojno učenje sta VidTIMIT in DeepfakeTIMIT. VidTIMIT vsebuje nabor originalnih posnetkov in je uporabljen kot pozitiven vzorec, DeepfakeTIMIT vsebuje ponarejene posnetke in je uporabljen kot negativen vzorec. DeepfakeTIMIT vsebuje 2 vrsti posnetkov. Prva vrsta posnetkov je v nizki ločljivosti, druga pa v visoki ločljivosti. Rezultate metode za odkrivanje artefaktov podrobneje analiziramo v poglavju Rezultati.

5.2 Implementacija

Implementacija metode za zaznavanje artefaktov je izvzeta iz članka [5]. Za treniranje CNN modela lahko simuliramo napake v ločljivosti slike, ki nastanejo pri afinih transformacijah obraza. Najprej je treba v sliki prepoznati obraz, iz katerega lahko nato pridobimo obrazne značilke. Nato izračunamo transformacijsko matriko, s katero lahko poravnamo vse obraze na isto standardno konfiguracijo. Ko imamo poravnan obraz na njem izvedemo Gaussovo zameglitev, s katero simuliramo napake, ki bi se zgodile ob kreiranju globokih ponaredkov. Nato z uporabo inverzne transformacijske matrike spremenjeno sliko povrnemo nazaj v originalno obliko. Da še bolj povečamo raznolikost učnih podatkov, slikam spremenimo svetlost, kontrast, popačenje in ostrino. Za simulacijo več različnih primerov ločljivosti sliko poravnamo in preoblikujemo v različne velikosti (slika 5). S tem povečamo raznolikost podatkov.



Slika 5: Simuliranje različnih velikosti slike [5]

Treniranje modela globokih ponaredkov za generiranje lažnih slik, ki bi služili kot negativni vzorci pri učenju našega modela, je zelo zamudno in zahteva veliko računskih virov. Metoda, ki je bila opisana v prejšnjem odstavku in ki jo tudi mi uporabljamo za generiranje negativnih vzorcev, je dosti hitrejša in enostavnejša, saj uporablja le preproste operacije za manipulacijo slik. Poleg tega je v primerjavi z drugimi metodami dosti bolj robustna, saj imajo nekatere druge metode težavo s prekomernim prileganjem učnim podatkom. Robustnost metode izhaja iz dejstva, da je iskanje artefaktov splošno med vsemi globokimi ponaredki, ne glede na metodo generiranja globokih ponaredkov.

Za učenje nevronske mreže uporabimo predele slike imenovane "regions of interest" (RoI). To so regije med obrazom in ostalim delom slike, pri katerih najpogosteje prihaja do artefaktov. Regije so izbrane v obliki pravokotnika med obrazom in okolico. Natančneje RoI določimo s formulo:

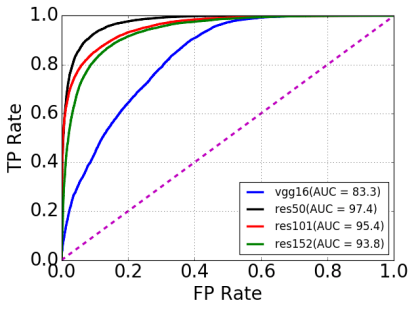
$$[y_0 - \hat{y}_0, x_0 - \hat{x}_0, y_1 + \hat{y}_1, x_1 + \hat{x}_1]$$

Pri kateri spremenljivke y_0, x_0, y_1, x_1 določajo najmanjše območje b, ki pokrije celoten obraz brez obrobja lic. Spremenljivke $\hat{y}_0, \hat{x}_0, \hat{y}_1, \hat{x}_1$ imajo naključne vrednosti iz območja $[0, \frac{h}{5}]$ in $[0, \frac{w}{8}]$, kjer je h višina in w širina območja b.

Za odkrivanje artefaktov uporabljamo 4 različne modele, ki se razlikujejo po načinu učenja in optimizacijski metodi. Razlikujemo med VGG16 [8], ResNet50, ResNet101 in ResNet152 [3]. Pri VGG16 uporabljamo optimizacijsko metodo SGD in model treniramo do 100 ponovitev. Ostali modeli uporabljajo že natreniran model ImageNet, ki ga nato le prilagodimo svojim podatkom. Zato za treniranje teh modelov uporabimo do 20 ponovitev. Med seboj se razlikujejo po številu nivojev. ResNet50 ima 50 nivojev, ResNet101 ima 101 nivojev in ResNet152 ima 152 nivojev nevronske mreže.

5.3 Rezultati

Rezultati naučenih modelov so kar presenetljivi. Uspešnost modelov lahko vizualiziramo z uporabo ROC krivulje in vrednosti AUC. ROC krivulja prikazuje točnost modela pri različnih klasifikacijskih pragih. AUC vrednost predstavlja ploščino pod krivuljo. Visok AUC pomeni, da model naredi manj napak pri razvrščanju. Uspešnost modela pri uporabi nabora podatkov iz UADFV je prikazana na sliki 6.

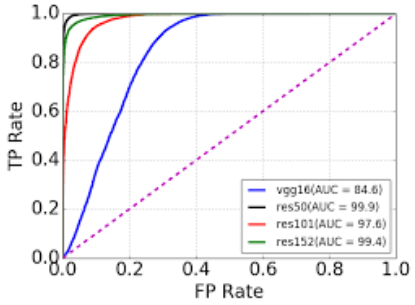


Slika 6: ROC krivulja za nabor podatkov iz UADFV [5]

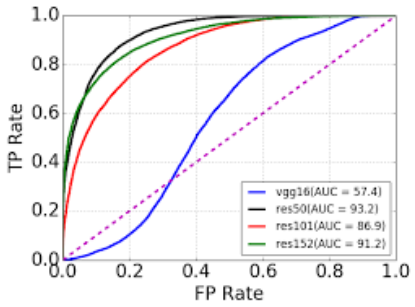
Ker so posnetki iz nabora podatkov VidTIMIT in DeepfakeTIMIT vsebovali 2 različni vrsti posnetkov, in sicer posnetke v nizki in posnetke v visoki ločljivosti, so rezultati prikazani na dveh slikah. Rezultati posnetkov v nizki ločljivosti so prikazani na sliki 7 in rezultati posnetkov v visoki ločljivosti so prikazani na sliki 8.

V primerjavi z drugimi metodami za odkrivanje globokih ponaredkov se opisani modeli dosti boljše obnesejo. Algoritme smo primerjali s Two-stream NN [11], Meso-4, MesoInception-4 [1] in HeadPose [9]. Uspešnost algoritmov lahko razberemo iz spodnje tabele 1.

Iz tabele 1 lahko razberemo, da je najuspešnejši model ResNet50, s kar 97,4 odstotno uspešnostjo na podatkih iz UADFV in 99,7, ter 93,2 odstotno uspešnostjo na podatkih iz DeepfakeTIMIT. Največja razlika v primerjavi z drugimi



Slika 7: ROC krivulja za posnetke nizke kakovosti iz DeepfakeTIMIT [5]



Slika 8: ROC krivulja za posnetke visoke kakovosti iz DeepfakeTIMIT [5]

Methods	UADFV	DeepfakeTIMIT	
		LQ	HQ
Two-stream NN	85,1	83,5	73,5
Meso-4	84,3	87,8	68,4
MesoInception-4	82,1	80,4	62,7
HeadPose	89,0	-	-
VGG16	84,5	84,6	57,4
ResNet50	**97,4**	**99,9**	**93,2**
ResNet101	95,4	97,6	86,9
ResNet152	93,8	99,4	91,2

Tabela 1: Primerjava AUC vrednosti z drugimi metodami za detekcijo globokih ponaredkov [5]

modeli se opazi na posnetkih v visoki ločljivosti, saj je model ResNet50, za skoraj 20 odstotkov boljši od najboljšega modela (Two-stream NN) izmed primerjanih modelov. Prednost ResNet modela je ta, da se ne prilagodi preveč na učne podatke in je zato boljši v primerjavi z ostalimi modeli.

6. ZAKLJUČEK

Tehnologija globokih ponaredkov je že tako napredovala, da je posnetke težko ločiti s prostim očesom. Kljub temu ustvarjeni posnetki vsebujejo napake oziroma artefakte, ki povedo, da posnetek ni resničen. To so večinoma napake v senčenju, odsevu ali modeliranju podrobnosti na obrazu. Za njihovo zaznavanje ponavadi uporabljamo nevronske mreže.

Metoda za razpoznavanje, ki smo jo opisali, razloči posnetke globokih ponaredkov od resničnih na podlagi razlik v resoluciji med obrazom in okolico. Do teh razlik pride zaradi afinih transformacij med ustvarjanjem posnetka. Glavna prednost te metode je, da za generiranje nabora podatkov ni potrebno treniranje modela globokih ponaredkov, ker lahko negativne primere ustvarimo z osnovnimi transformacijami resničnih slik. To poveča tudi robustnost predstavljene metode. Rezultati te metode presežejo rezultate drugih metod.

7. LITERATURA

[1] D. Afchar, V. Nozick, J. Yamagishi, and I. Echizen. Mesonet: a compact facial video forgery detection network. In *2018 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–7. IEEE, 2018.

[2] V. Conotter, E. Bodnari, G. Boato, and H. Farid. Physiologically-based detection of computer generated faces in video. In *2014 IEEE International Conference on Image Processing (ICIP)*, pages 248–252. IEEE, 2014.

[3] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[4] Y. Li, M.-C. Chang, and S. Lyu. In ictu oculi: Exposing ai generated fake face videos by detecting eye blinking. *arXiv preprint arXiv:1806.02877*, 2018.

[5] Y. Li and S. Lyu. Exposing deepfake videos by detecting face warping artifacts. *arXiv preprint arXiv:1811.00656*, 2018.

[6] F. Matern, C. Riess, and M. Stamminger. Exploiting visual artifacts to expose deepfakes and face manipulations. In *2019 IEEE Winter Applications of Computer Vision Workshops (WACVW)*, pages 83–92. IEEE, 2019.

[7] N. Rahmouni, V. Nozick, J. Yamagishi, and I. Echizen. Distinguishing computer graphics from natural images using convolution neural networks. In *2017 IEEE Workshop on Information Forensics and Security (WIFS)*, pages 1–6. IEEE, 2017.

[8] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[9] X. Yang, Y. Li, and S. Lyu. Exposing deep fakes using inconsistent head poses. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8261–8265. IEEE, 2019.

[10] H. Zhao, W. Zhou, D. Chen, T. Wei, W. Zhang, and N. Yu. Multi-attentional deepfake detection. *arXiv preprint arXiv:2103.02406*, 2021.

[11] P. Zhou, X. Han, V. I. Morariu, and L. S. Davis. Two-stream neural networks for tampered face detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1831–1839. IEEE, 2017.

On Learning to Detect Manipulated Facial Images

Metodija Bucevski
Faculty of Computer and Information Science, Večna pot 113
Ljubljana, Slovenia
mb8696@student.uni-lj.si

Matej Rus
Faculty of Computer and Information Science, Večna pot 113
Ljubljana, Slovenia
mr0022@student.uni-lj.si

Jakob Bernik
Faculty of Computer and Information Science, Večna pot 113
Ljubljana, Slovenia
jb9255@student.uni-lj.si

ABSTRACT
The paper describes the realism of image manipulation, and how difficult is to detect them automatically or by humans. With the rapid development of image generation and manipulation there is a need for a system to detect manipulated images. Manipulated images can be used for spreading false information, identity stealing etc. They created automated benchmark for facial manipulation detection for easy comparing.

Keywords
Image manipulation, forgery detection methods, face forensics

1. INTRODUCTION
In today's age of globalization, the issue of the authenticity of digital information is becoming an increasingly pressing issue. The development of tools and methods that make it possible to alter and falsify, in particular, visual information such as images and videos that people are more inclined to trust, allows individuals to spread false news easier or even impact public opinion. A special place here is occupied by the manipulation of body and face language, as in this way we can, for example, influence the opinion of the general public by impersonation of an important and well-known person. It is therefore not surprising that the need for tools to successfully detect and prevent the spread of such information is growing.

Basically, the methods of facial manipulation are divided into two categories as we can see on figure 1: Control of facial expressions and control of identity. One of the well-known methods for controlling facial expressions is called *Face2Face*. It allows transfer of the desired facial expressions from the original face to another person. The second category, instead of facial expressions, replaces a person's entire face, which is why we also call it a face replacement. One of the methods that falls into this category is called *DeepFakes*. In its operation, it uses the principles of deep learning, and consequently leads to better but more time-consuming results.

In this paper we discuss and present advances in the field of automatic detection for facial manipulations as presented in [4]. Compared to human observer, the presented approaches improve the detection of fake ones by a significant margin.

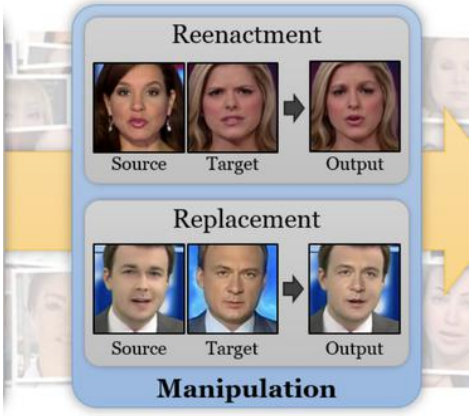


Figure 1: Different approaches to facial manipulation.

2. RELATED WORK
The content of this paper touches different fields and studies on facial expressions, digital forensics and computer vision. Some of them are presented in this section.

2.1 Facial expressions in social interactions
In the article [3], the author focuses on our perception of facial expressions and the impact that information can have on an individual's behavior. Certain individual behaviors can elicit rapid responses that often reflect the emotions of the observed person. In certain cases, this can even happen so that the observer is not even aware of it. It is becoming increasingly clear that much of the information processing in the brain happens unconsciously. Only then does information become accessible to consciousness, and in many cases, unconscious processes can also control our behavior and responses. Studies show that we are less aware of our surroundings than we think. We are able to respond accurately and quickly to a particular stimulus even before we

are aware of it. The study of facial expressions shows very well how behavior can quickly change into a refined communication system. A certain person can send unconscious signals to the viewer in a sophisticated way through modern communication media and correct facial expressions, thus influencing his behavior.

2.2 Exploiting Spatial Structure for Localizing Manipulated Image Regions

In this article [1] they are describing problem of fast development high-tech tools for manipulating images in the way that can easily avoid state-of-the-art image detection approaches.The recent success of deep learning approaches in different recognition tasks inspired them to develop a high confidence detection framework which can localize manipulated regions of an image. They formulated a hybrid CNN-LSTM model for capturing discriminative features between manipulated and non-manipulated regions. Their motivation was to learn the boundary of the spatial structure, between manipulated and non-manipulated regions with the combination of LSTM and convolutional layers. Developed framework was able to detect different types of image manipulations such as copy-move, removal and splicing. Implemented detection framework is capable of locating manipulated patches as well as pixel level. Most of the existing forgery detectors works by focusing on specific methods such as copy-move, splicing. Image tamper detection should be able to detect all types of manipulation other than focusing on specific type.

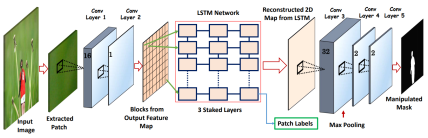


Figure 2: Overview of proposed framework for joint tasks-patch classification and manipulated region segmentation.

2.3 Automatic detection based on Learned Features

Figure 2 shows the general overview of how manipulated images can be detected. More specifically, there is an input image that may or may not be fake. This image is not directly forwarded to the input of the classification process, instead, first face tracking method is used to locate the face area. The area is extracted and forwarded as an input to the classificator. Output of the classification process is binary value, i.e. fake or not fake input image. When it comes to choosing classificator, authors[4] decided to use Xception method.

Xception, also called Extreme Inception is convolutional neural network which architecture is linear stack of depthwise separable convolution layers with residual connections. This convolution consists of pointwise convolution followed by a depthwise convolution, as shown in figure 3. First, we examine how this convolutions work.

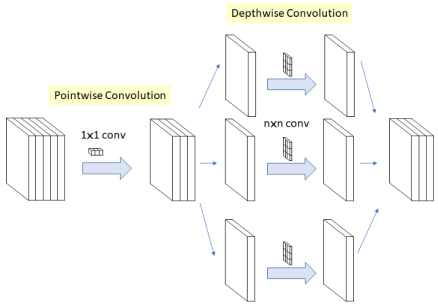


Figure 3: Depthwise Separable Convolution

Pointwise convolution means 1x1 convolution for changing the dimension, i.e. changing the number of channels for each image. It uses 1x1 kernel, which means this kernel iterates through every single point and has depth equal to the number of channels of the image. The second kernel is used for depthwise convolution which means channel-wise nxn spatial convolution. Spatial convolution is defined on a spatial variable rather than time variable. The main idea why depthwise separable convolution is used rather than the ordinary convolution is because the number of operation (multiplications) are lot less and the reason why spatial separable convolutions is not used is because not all kernels can be factored into two smaller kernels.

Its predecessors used first depthwise convolution followed by pointwise convolution. Because of this, in the original Inception Module, after first operation there is non-linearity, whereas in Xception there is no intermediate ReLU non-linearity. This is because experimentations showed Xception without any intermediate activation has the highest accuracy compared with the one using ELU or ReLU, as shown in Figure 4.

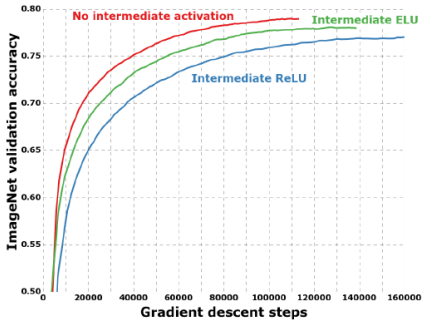


Figure 4: Depthwise separable convolution with different activation units

Exponential Linear Unit (ELU) is similar to ReLU except it can have negative inputs. They are identity function for non-negative inputs. ReLU for negative inputs is 0, whereas ELU can have negative values up till $-\alpha$, to which it converges. The purpose of these activation units is to increase the number of parameters of a CNN. But if the network is small, and the advantage of depthwise separable convolution is to decrease the number of parameters, then the

model will have small number of parameters which translates into failure for the network to learn properly during training. Authors[4] concluded from their experiments, that Xception architecture with 14 modules consists of enough parameters. More precisely 22,855,952, whereas Inception V3 that uses intermediate activation units has a little bit more, 23,626,728.

Xception architecture has 36 convolutional layers that form the feature extraction base of the network. They are structured into 14 modules that on figure 6 are represented as SeperableConv which is the depthwise separable convolution explained previously. Each has linear residual connections around them, except the first and the last module.By residual connection we mean shortcut/skip connections for increasing the accuracy. The impact of having residual connections on the accuracy versus not having is shown on figure 5. This architecture is easy to define and modify.

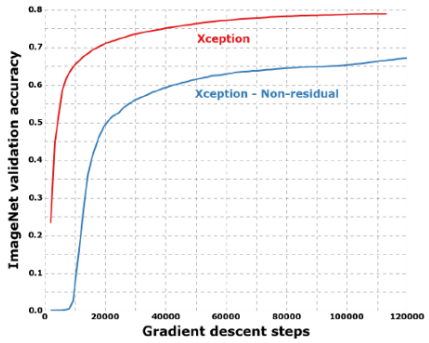


Figure 5: Skip connection's impact on accuracy

Let's say that the input image, also called tensor, is with dimensions of 299x299x3. First two modules are normal convolutions. The output of each convolution are 728 feature maps with dimensions 19x19. There is additional operation called MaxPooling, that calculates the maximum presence of a feature in each and every patch of the feature map. The final output, i.e. the output from the exit flow are 2048-dimensional vectors that are multiplies with weight matrix to obtain fully-connected layer. Finally, logistic regression is used. Logistic regression is used to find relationship between one or more independent variables (in our case features) and binary dependent variable(in our case with value "fake" or "not fake").

The key feature, when it comes to Xception is that mapping of cross-channel and spatial correlations in the feature maps of CNN can be entirely decoupled. The data first goes through the entry flow, then 8 times through the middle flow and at the end through the exit flow. All Separable Convolution layers use a depth multiplier of 1, this means no depth expansion.[2]

Figure 7, presents a Xception accuracy for detecting facial forgery with respect to different video qualities (raw, HQ, LQ), number of training corpus and manipulation method (Deepfakes, Face2Face, FaceSwap, NeuralTextures and all combined).

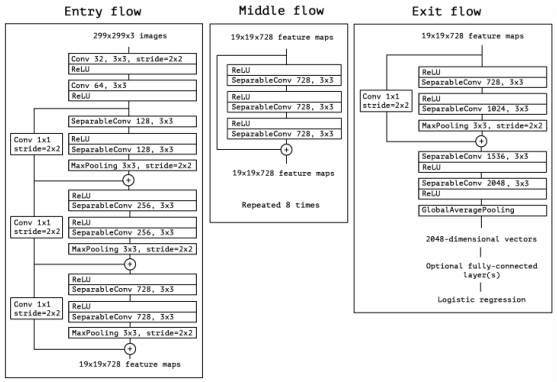


Figure 6: Overall Architecture of Xception

From the graphs, authors[4] came to the following conclusions:

- Detection performance drops when compression is higher.
- Detection performance increases with size of training corpus, especially for Low Quality(LQ) videos.

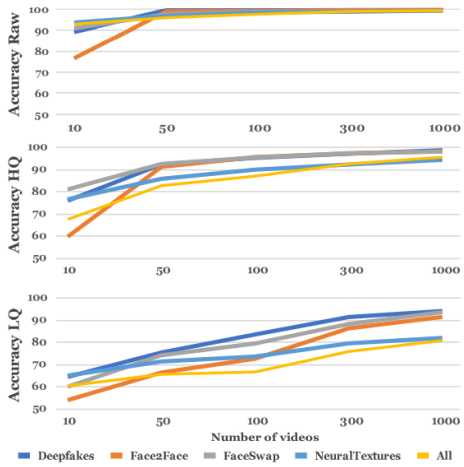


Figure 7: Xception detection accuracy

3. LARGE-SCALE FACIAL FORGERY DATABASE

A core contribution of original paper[4] is their FaceForensics++ dataset extending the preliminary FaceForensics dataset. It enables them to train forgery detector for facial image manipulation. Most of their images is from YouTube videos. For their data set they choosed two computer based approaches (Face2Face and FaceSwap) and two learning based approaches (DeepFakes and NeutralTextures).

3.1 FaceSwap

Graphics based approach to transfer the face region from a source video to target video. The face region Is extracted based on sparse detected facial landmarks. This landmarks are then used to fit 3D template model using blendshaped. Model is then back-projected to the image by minimizing the difference between the projected shape and localized landmarks using the textures of the input image. The computation is lightweight and can be executed fast on average CPU.

Figure 8: Examples of final output for faceSwap method.

3.2 DeepFakes

Face replacement based on deep learning. The term DeepFakes comes from creating fakes using deep learning. A face in target system is replaced by face from a source video or image collection. Method works by two auto encoders with a shared encoder that are trained to reconstruct training image of the source and the target face. To crop and align images they used face detector. For creating fake images the trained encoder and decoder of the source face are applied to the target face.

Figure 9: DeepFake method manipulation example.

3.3 Face2Face

System that can transfer facial expression of a source video to a target video while maintaining the identity of the target person. We have 2 video streams with manual key-frame selection. Frames can be used to generate dense reconstruction of the face by re-synthesize the face under different illuminations and expressions. The example can be seen in figure 10.

Figure 10: Example of face2face manipulation.

3.4 Neural Textures

Neural networks can be utilized to synthesize artificial 2D imagery, like generative adversarial networks (GANs) and auto-regressive networks do. Both of them show excellent results when synthesizing single, isolated image, but the problem arises when synthesizing 3D and temporally-consistent image.

Neural textures combine 3D transformations and computer graphic pipeline with learnable rendering, also known as Deferred Neural Renderer. Before continuing, we need to define what are Neural Textures and what is Deferred Neural Renderer.

Neural Textures are learned feature maps that are trained as part of the scene capture process and contain more information, than regular textures. A texture contains several information such as the albedo of an object, i.e. the ability to reflect light, high-frequency geometric detail in the form of normal or displacement maps and etc. Neural textures can have arbitrary dimensions and store high-dimensional learned feature vectors per texel. Texel also called texture pixel is basic unit of texture space similar to pixel in a picture. So an array of texels create a texture. Whereas texture is application of a type of surface to a 3D image. This surface can be for example uniform or irregular like marble, as shown on figure 11. This surface is wrapped around the 3D object.

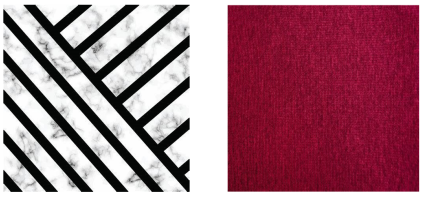


Figure 11: Irregular and uniform texture

Authors[4] used standard graphic pipeline, for sampling neural textures. This pipeline is also called rendering pipeline which is a model that describes what steps a graphics system needs to perform to render a 3D scene to a 2D screen, i.e. how to present 3D object on a screen. It can be divided into three main parts: Application where changes are made to the scene(input from the user), geometry which deals with almost all operations with polygons and their vertices and rasterization which means converting an image described in a vector graphics format to raster image, that is series of pixels that when displayed together create the image.

The problem with this pipeline is that it can synthesize images with high visual quality only if on its input it was sent well-defined, high-quality 3D content. This problem is solved by using Deferred Neural Renderer, which is the same pipeline together with learnable components. This means its stages can be made learnable. So the key components are Neural Textures and Deferred Neural Renderer, and using both of them, one can synthesize images where the original 3D content was imperfect and also manipulate existing videos in static and dynamic environments at real-time rates – facial reenactment.

Facial reenactment, that can be seen on figure 12, means to transfer facial expressions of one person to another person in real time. It requires only a few milliseconds for high-resolution output, which is significantly faster than traditional reenactment. For changes in the eye region, it is necessary to have additional conditional input for the eye movement in the rendering network. Also, from several experiments, it was concluded that Neural Texture approach is one of the most difficult method for the human eye to detect whether the image is original or faked. The reason for this is that this method does not introduce strong semantic change.

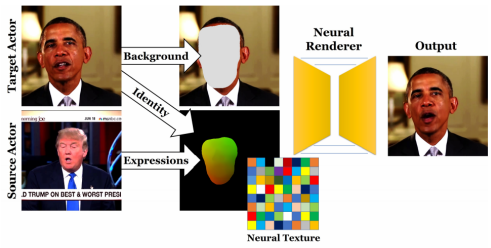


Figure 12: Facial reenactment

It is important to choose right texture resolution for a given scene and viewpoint. If the scene is complex, i.e. with high depth complexity, there will be more optimal choices. The problem is that the parts of the scene that are far from the viewer will be under-sampled, and those that are near the viewer will be oversampled. The first is referred as minification, and the latter is referred as magnification. Both can happen in the same scene, just in different parts.

For elimination of this problem, Neural Texture Hierarchies are used with K levels. To access this hierarchy, sampling is performed from all K levels using bi-linear sampling and normalized texture coordinates. All per-level sampling results are added together, to get the final color estimate. The end result should be storing all low frequency information on the coarse levels and high frequency information on finer levels. To do that, authors[4] applied increasing amount of a L2 regularization to the channels of the finer levels and no regularization on the coarse levels. The benefit here is that if only one high resolution neural texture is used to solve the problem with minification, another problem arises called overfitting due to texture magnification. Texture magnification means sampling unoptimized pixel.

There are several constraints while using this approach. This approach relies on a geometry proxy that has to be reconstructed in a preprocessing step. If the geometry is too coarse, the result quality degrades, i.e., the re-rendered images get more blurry. As the appearance of each object is unique, it is necessary to train the neural texture for every new object and it is assumed static illumination, meaning it is not possible to relight the scene. This approach can also be used for cloning and removal of multiple objects in a , as shown on figure 13.[5]

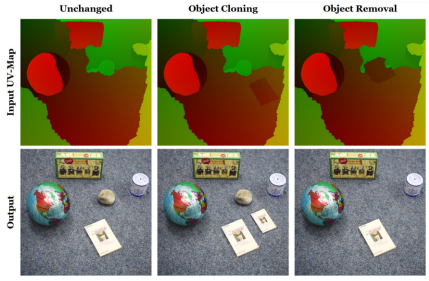


Figure 13: Multiple object scene - removal and cloning

4. DETECTION RESULTS

The creators of the FaceForensics++ dataset conducted an interesting survey to measure how well people are at detecting facial manipulations in images. The participants of this test were students with background in computer science and final results of this tests are presented in figure 14.

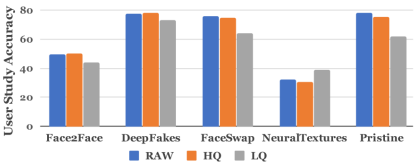


Figure 14: Results for human detection of manipulations in images by methods and image quality, conducted on FaceForensics++ dataset.

We can observe that humans don't perform well at detecting facial expression manipulation methods, that being face2face and neural textures. For neural textures in particular, the results proved that we are better of with guessing, if some image is manipulated or not, rather then looking for some indicators to base our decision on.

The automatic detection test was performed using different learning models and different qualities of the test image. The results are presented in figure 15.

Compression	Raw	HQ	LQ
[14] XceptionNet Full Image	82.01	74.78	70.52
[27] Steg. Features + SVM	97.63	70.97	55.98
[17] Cozzolino *et al.*	98.57	78.45	58.69
[10] Bayar and Stamm	98.74	82.97	66.84
[51] Rahmouni *et al.*	97.03	79.08	61.18
[5] MesoNet	95.23	83.10	70.47
[14] XceptionNet	**99.26**	**95.73**	**81.00**

Figure 15: Detection results for different learning models on different image qualities for the FaceForensics++ dataset.

From the figure 15, we can observe that XceptionNet based learning model performs best for all image qualities. The images used were all cropped around the face, with exception

of XceptionNet Full image which took as an input the whole image.

Together with the database, a measurement system for detection of facial forgery was also presented in the original paper [4]. This benchmark is publicly available and anyone can submit their trained models to test how good are they at forgery detection. The images used for this test are taken from 1000 additionally pre-processed videos to ensure as realistic conditions as possible. The results for creator's own model, trained on low quality images are presented in the figure 16.

Accuracies	DF	F2F	FS	NT	Real	Total
Xcept. Full Image	74.55	75.91	70.87	73.33	51.00	62.40
Steg. Features	73.64	73.72	68.93	63.33	34.00	51.80
Cozzolino *et al.*	85.45	67.88	73.79	78.00	34.40	55.20
Rahmouni *et al.*	85.45	64.23	56.31	60.07	50.00	58.10
Bayar and Stamm	84.55	73.72	82.52	70.67	46.20	61.60
MesoNet	87.27	56.20	61.17	40.67	**72.60**	66.00
XceptionNet	**96.36**	**86.86**	**90.29**	**80.67**	52.40	**70.10**

Figure 16: Results for each detection method in benchmark. Summarized after [4].

Again we can see that XceptionNet based model performs best overall, with exception of MesoNet being better at distinguishing non-manipulated images.

5. CONCLUSION

In this paper we focused on different ways how one can manipulate digital content such as images and videos, for it's own benefit. The most important ones are divided into two groups, based on what we change. We either change whole face, or just the expression of the subject. We also divide them in additional two groups, based on using deep learning or using computer graphic approach. In this sense, FaceSwap is graphic based, identity manipulation approach, DeepFakes is deep learning, identity manipulation approach, Face2Face is graphic based, facial expression manipulation approach and Neural textures are deep learning, facial expression manipulation aproach. Although some methods achieve very good results, there are also ways to disclose them. Learning models for detecting counterfeits can achieve better results as opposed to human observer, especially on lower-quality videos and images, where people tend to struggle. As the forgery methods develop through time, so must the ones that detect the manipulations happening, and the FaceForensics++ dataset offer us new means to battle the increasing amount of fake information on the internet.

6. REFERENCES

[1] J. H. Bappy, A. Roy-Chowdhury, J. Bunk, L. Nataraj, and B. S. Manjunath. Exploiting spatial structure for localizing manipulated image regions. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 4980–4989, 2017.

[2] F. Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[3] C. Frith. Role of facial expressions in social interactions. *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*, 364:3453–8, 12 2009.

[4] A. Rössler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Nießner. Faceforensics++: Learning to detect manipulated facial images. *CoRR*, abs/1901.08971, 2019.

[5] J. Thies, M. Zollhöfer, and M. Nießner. Deferred neural rendering: Image synthesis using neural textures. *ACM Transactions on Graphics (TOG)*, 38(4):1–12, 2019.

Generični algoritem za klesanje metapodatkov na podlagi časovnih žigov

Opis in ponovna izvedba eksperimenta

Alen Bizjak
Fakulteta za računalništvo in informatiko
Univerza v Ljubljani

Matevž Špacapan
Fakulteta za računalništvo in informatiko
Univerza v Ljubljani

Tine Fajfar
Fakulteta za računalništvo in informatiko
Univerza v Ljubljani

POVZETEK

Na področju digitalne forenzike ima veliko vlogo klesanje datotek. To je postopek pridobivanja datotek iz medija, o katerem nimamo podatkov, kje se datoteke nahajajo. Taki postopki so več ali manj uspešni pri pridobivanju samih datotek, redko pa se osredotočajo na metapodatke, povezane z njimi. Avtorji članka [6] so razvili nov generični algoritem za klesanje metapodatkov na podlagi časovnih žigov in ga testirali na datotečnem sistemu NTFS in ext4. Algoritem se je dobro izkazal, saj je našel vse datoteke na poškodovanem NTFS disku.

Ključne besede

metapodatki, klesanje, digitalna forenzika

1. UVOD

V računalništvu se vsakodnevno srečujemo z manipulacijo datotek, ki jih imamo shranjene na raznoraznih medijih. Na prvi pogled se nam lahko sicer zdi, kot da se ta postopek izvaja enako na različnih napravah, vendar temu ni tako. Ko datoteko zapišemo na medij moramo shraniti še nekaj dodatnih podatkov (tj. metapodatkov) o njej, kot so lokacija, ime datoteke, čas vnosa ipd. Brez teh dodatnih atributov bi bilo iskanje datotek na mediju izjemno počasno, morda tudi nepravilno. Vprašanje je, kaj se zgodi z datoteko, ko jo izbrišemo: se je res vsebina uničila ali se je le označila kot, da ni prisotna?

1.1 Prispevek članka

V članku [6] se avtorji spopadejo z novim načinom iskanja izbrisanih datotek na mediju, s pomočjo klesanja metapodatkov. Predstavljen algoritem išče časovne žige, ki so shranjeni v metapodatkih datotek, da se hitro določi potencialne lokacije datotek. Algoritem to počne tako, da išče zaporedne ponavljajoče se vzorce, ki bi lahko predstavljali časovne žige. Vse zadetke se nato preišče še s skripto za evalvacijo, saj se lahko v prvem algoritmu znajdejo tudi naključna ujemanja, ki se jih s skripto izloči.

1.2 Struktura članka

V poglavju 2 najprej predstavimo področje interesa. Osredotočimo se tudi na delovanje datotečnih sistemov, ter bolj konkretno opišemo implementacijo NTFS ter ext4 datotečnih sistemov. Opišemo tudi kaj pomeni pojem klesanje ter kako se ločita klesanje datotek od klesanja metapodatkov. V poglavju 3 opišemo delovanje predstavljenega generičnega algoritma ter evalvacijo zadetkov le-tega, nato v poglavju 4 opišemo izvedbo eksperimenta uporabe predstavljenega algoritma na NTFS ter ext4 datotečnih sistemih. V poglavju 5 predstavimo rezultate iz članka [6] ter rezultate, ki smo jih pridobili sami s ponovitvijo eksperimenta. V poglavju 7 povzamemo najdbe ter podamo svoje zaključne misli.

2. PREGLED PODROČJA

Za pravilno shranjevanje datotek na medij je odgovoren datotečni sistem, ki ga določimo ob formatiranju medija. Datotečni sistemi določajo lokacijo shranjevanja na mediju ter vse metapodatke povezane s to datoteko. Večina jih hrani isti osnovni nabor podatkov, kot so ime, lokacija na mediju, pravice dostopa ter nekaj časovnih žigov. Kateri datotečni sistem bomo uporabljali je mnogokrat pogojeno z operacijskim sistemom, ki ga naprava poganja. Tako se na OS Windows srečamo z datotečnimi sistemi kot so FAT ali NTFS, medtem ko na sistemih, ki temeljijo na Unix jedru, najdemo datotečne sisteme kot so ext4, HFS+, UFS, itd.

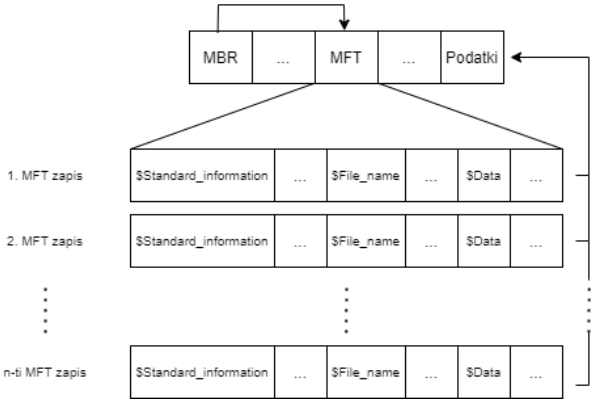
2.1 Datotečni sistemi

Če bi datotečne sisteme med seboj primerjali bi opazili, da se med seboj razlikujejo po tem kako in katere metapodatke shranjujejo (recimo kakšna je struktura indeksa), po njihovi varnosti, velikosti (ter s tem povezano omejitvijo velikosti medija), hitrosti, robustnosti itd. Pot do indeksne datoteke, z navedeno vsebino, je shranjena v tabeli particije, ki jo poznamo pod imenom Master Boot Record (MBR; glavni zagonski zapis). V tem članku se predvsem osredotočamo na metapodatke, ki hranijo čas povezan z nekim dejanjem nad datoteko. Tudi pri shranjevanju teh podatkov se najde nekaj razlik med datotečnimi sistemi, vendar pa so pri vseh skupni sledeči časovni žigi: čas ustvarjanja datoteke, čas spreminjanja vsebine ter čas zadnjega dostopa.

2.1.1 *Datotečni sistem NTFS*

Z NTFS datotečnim sistemom se primarno srečujemo ob uporabi OS Windows, saj je to privzeti način formatiranja diska ob njegovi namestitvi. Datotečni sistem shranjuje svoje metapodatke v datoteki *Master File Table* (MFT, tj. glavni tabeli datotek) [5]. Vse spremembe indeksa se shranjuje v posebno datoteko `$LogFile`, obstaja pa tudi datoteka `$MFTMirr`, ki služi kot (delna) varnostna kopija originalne datoteke, saj lahko z njo do neke mere obnovimo vsebino MFT. Na sliki 1 lahko vidimo poenostavljen postopek iskanja datoteke na mediju ter nekaj izpostavljenih atributov v posameznih zapisih znotraj indeksa: ob priklopu medija oz. zagonu sistema ugotovimo lokacijo MFT indeksa, v njem pa nato najdemo še lokacijo iskane datoteke.

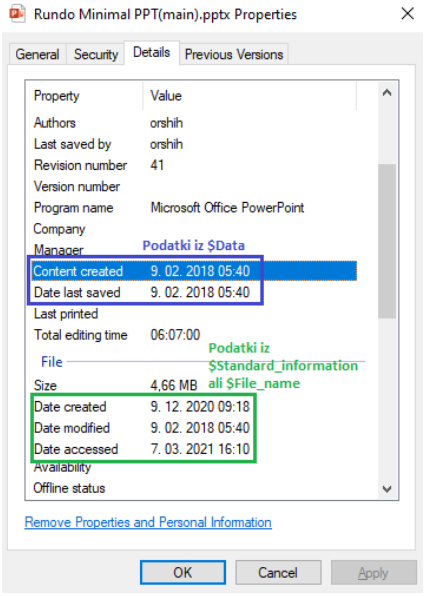
Slika 1: Zgradba in povezave v NTFS datotečnem sistemu.



Metapodatki so shranjeni v atributih znotraj posameznega MFT zapisa, izmed katerih so trije takšni, kjer se lahko nahajajo tudi časovni žigi [9]. Poleg treh časovnih žigov opisanih v poglavju 2.1, se hrani še čas zadnje spremembe zapisa v MFT tabeli. Čas je predstavljen s 64-bitnimi nepredznačenimi števili z natančnostjo 100ms v UTC formatu, beleži pa se ga od 1.1.1601 00:00 dalje kot število preteklih enot časa. Atributi, ki so polje našega interesa so:

- `$STANDARD_INFORMATION` v katerem so poleg štirih časovnih žigov shranjeni še podatki o pravicah dostopa do datoteke. Časovne žige lahko v tem atributu spreminjajo uporabniki medija (npr. z uporabo posebnih aplikacij),
- `$FILE_NAME`, ki poleg štirih časovnih žigov hrani še dodatne podatke o datoteki (njena velikost, zastavice, ime ter mapo v kateri se datoteka nahaja). Časovne žige v tem atributu lahko spreminja le sistemsko jedro,
- `$DATA`, ki pa nima predpisane vsebine ali strukture, vendar lahko vanj aplikacija shrani poljubne metapodatke, med katerimi so lahko tudi časovni žigi (primer na sliki 2).

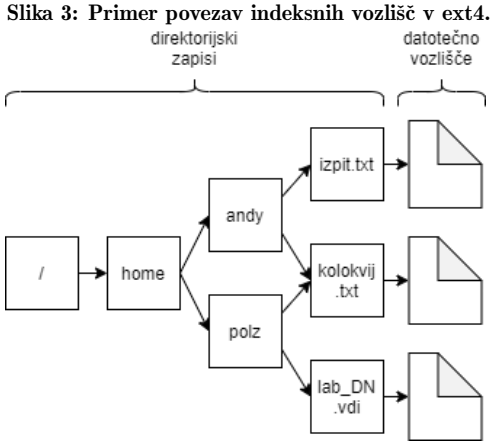
Slika 2: Metapodatki datoteke.



2.1.2 *Datotečni sistem ext4*

ext4 (ter prejšnji različici ext2 ter ext3) datotečni sistem se primarno uporablja v operacijskih sistemih z jedrom Linux. V njem se shranjuje metapodatke v t.i. indeksnih vozliščih (*inode* oz. index node). Za namen članka je dobro izpostaviti še to, da se poleg treh časovnih žigov iz poglavja 2.1 hrani v vozliščih še en, ki nam pove kdaj je bilo le-to izbrisano, v kolikor se je ta dogodek zgodil. Časovni žigi so shranjeni kot 64-bitna števila, vendar v dveh delih: 34 bitov, ki povejo število preteklih sekund od leta 1.1.1970 00:00, dodatnih 30 bitov pa omogoča natančnejše hranjenje časa, do nanosekunde natančno. V primerjavi z NTFS, se način indeksiranja razlikuje v tem, da lahko določeno vozlišče kaže na drugo vozlišče, namesto na lokacijo datoteke na mediju. To pomeni, da korensko vozlišče "\" kaže na vse mape, ki so direktno podrejene le-temu, nato se postopek ponavlja še v teh mapah itd., vse dokler ne pridemo do končnega vozlišča (grafičen primer na sliki 3) [2].

Slika 3: Primer povezav indeksnih vozlišč v ext4.



Indeksna vozlišča ločimo v dve kategoriji:

- **direktorijski zapis**, katerih vsebina je lahko mapa ali ime datoteke (predstavljamo si, kot da ime datoteke predstavlja mapo). V teh zapisih najdemo kazalce, ki kažejo na druga vozlišča, če zapis predstavlja mapo, oziroma na datotečni zapis, če ta predstavlja ime datoteke,
- **datotečno vozlišče**, ki hrani metapodatke vezane na datoteko ter lokacijo datoteke na mediju.

Izkaže se, da niso vsi podatki o datoteki shranjeni v končnem vozlišču, temveč se lahko nahajajo tudi znotraj zadnjega direktoriskega zapisa. Primer takšnega podatka je ime datoteke, saj lahko z navideznimi povezavami v sistemih Linux več direktorijskih zapisov usmerimo na lokacijo istega datotečnega vozlišča, s tem pa prihranimo prostor na mediju. Ko se na mediju povečuje količina indeksnih vozlišč, se tudi podaljšuje iskalni čas določene datoteke. Zato se vozlišča združuje v *bločne skupine*, ki so shranjene v *tabeli opisov skupin blokov*. Torej lahko na nivoju bloka ugotovimo, ali moramo iskanje nadaljevati v njem, ali pa ga lahko preskočimo. Dodatno pohitritev se doseže tudi z *bitno sliko* indeksnega vozlišča, ki za vsako polje v njem hrani ali je v uporabi ali ne.

2.2 Klesanje datotek

Postopek iskanja (izbrisanih) datotek na mediju imenujemo klesanje (angl. *carving*), pri tem pa se lahko osredotočamo na različne lastnosti datotek. V kolikor je vsebina indeksa datotek poškodovana, bomo lahko iskanje opravljali le nad podatki, ki so shranjeni na mediju. V teh primerih si lahko pomagamo s posebnimi vrednostmi (angl. *magic number*), ki so shranjene na začetku (ter koncu) datotek. Tako lahko recimo najdemo JPEG datoteko s posebno vrednostjo `0xFFD8` na začetku, ter `0xFFD9` na koncu. Določene datoteke ne bodo nujno vsebovale posebnih vrednosti, zaradi česar moramo v teh primerih iskanje opraviti nad morebitno vsebino datoteke [7].

2.3 Klesanje metapodatkov

Ugotovili smo, da se v vsakem datotečnem sistemu hrani indeks datotek, iskanje izbrisanih datotek z njegovo pomočjo pa je lahko poenostavljeno, saj se običajno ob brisanju zgolj spremeni zapis v indeksu – bodisi se ga izbriše ali, kot pokazano na primeru ext4, označi kot izbrisanega. Glede na vsebino indeksa lahko zmanjšamo potreben nabor iskanja podatkov le na dele medija, ki niso zabeleženi, saj za vse ostale lahko direktno ugotovimo vsebino. Za učinkovito klesanje je najbolje, da se zanašamo na metapodatke indeksa, ter lastnosti datotek iz prejšnjega podpoglavja [8].

3. OPIS ALGORITMA

Algoritem je sestavljen iz dveh delov: prvi del je algoritem za klesanje metapodatkov na podlagi časovnih žigov, ki vrne seznam potencialnih zadetkov. Drugi del so skripte za evalvacijo zadetkov, ki na podlagi specifike konkretnega datotečnega sistema filtrirajo zadetke.

3.1 Klesanje metapodatkov na podlagi časovnih žigov

3.1.1 Potrebne predpostavke

Prvi del algoritma temelji na predpostavki, da se v večini datotečnih sistemov časovni žigi posameznih datotek nahajajo fizično blizu ter da so enaki. Datoteki se ob ustvaritvi vsi časovni žigi postavijo na isto vrednost. V kolikor se datoteka v prihodnje ne spreminja, bo prva predpostavka zagotovljena. Druga predpostavka pa je povsem odvisna od arhitekture obstoječih datotečnih sistemov. Tega algoritma nikakor ne moremo uporabiti za datotečni sistem, ki časovnih žigov posamezne datoteke ne shranjuje fizično skupaj. Avtorji članka [6] so zbrali podatke o 11 datotečnih sistemih in s tem potrdili, da vsak od njih shranjuje po 3 ali več časovnih žigov fizično blizu. Časovni žigi so ponavadi shranjeni kot 32 ali 64 bitni zapisi na disku. Dolžina časovnega žiga v bitih je edini ne-generični parameter, ki ga prvi del algoritma potrebuje, saj je le-ta odvisen od arhitekture datotečnega sistema, iz katerega želimo klesati metapodatke.

Iz teh dveh predpostavk lahko ugotovimo, da v kolikor na disku najdemo niz bitov, kjer se nek 32 ali 64 bitni podniz ponovi dva ali večkrat v relativni bližini, to najverjetneje pomeni, da smo našli časovni žig neke datoteke. Lahko rečemo, da smo locirali njene metapodatke. Algoritem torej temelji na preprostem algoritmu za ujemanje nizov z drsečim oknom.

3.1.2 Implementacija

Naj bo m dolžina časovnega žiga v bajtih in naj bo T tabela bajtov, ki jo želimo preiskati. Naj bo k dolžina drsečega okna, v katerem želimo najti potencialno večkratno ujemanje časovnega niza. Glavna ideja algoritma je naslednja: vsako zaporedje dolžine m bajtov lahko smatramo kot časovni žig, nakar pogledamo naslednjih k bajtov in v njih iščemo ujemanje s prvimi m bajti. V kolikor najdemo ujemanje, javimo potencialno lokacijo metapodatka. Iskanje začnemo v $T[0]$ in za prvi časovni žig vzamemo $T[0:m]$. Nato pogledamo, ali se ta podniz nahaja nekje v $T[m:(m+k)]$. Zaradi hitrejšega izvajanja je originalna verzija algoritma zasnovana tako, da se drseče okno v primeru neujemanja naprej premakne za m bitov – s tem predpostavlja, da se vsi časovni žigi na disku nahajajo na večkratniku od m. V primeru, da smo našli eno ali več ujemanj, pa se drseče okno premakne za k. Ta implementacija dodatno zahteva tudi pogoj, da je k večkratnik od m, saj bi sicer zaradi neporavnanosti zgrešili mnogo časovnih žigov. V nadaljevanju podajamo psevdokodo algoritma.

Zaradi izločanja nesmiselnih zadetkov, kot so npr. ponavljajoče vrednosti 0x00...0 in 0xFF...F, je v algoritem vpeljana spremenljivka *ponavljajociBajti*, ki skrbi za filtriranje le-teh.

3.1.3 Časovna kompleksnost

Za evalvacijo časovne kompleksnosti si oglejmo najslabši možen scenarij: naj tabela T nima nobenih ponavljajočih časovnih žigov, ki bi jih naš algoritem lahko našel. Zaradi tega ne bomo nikoli premakniti drsečega okna za k; vedno le za m (k je večkratnik od m). Predpostavimo tudi, da ne filtriramo nobenih ponavljajočih bajtov. Torej bomo na vsakih m bajtov izvedli iskanje ujemajočih podnizov v drsečem oknu dolžine k. Iskalno proceduro bomo torej izvedli

Algoritem 1 Klesanje potencialnih časovnih žigov

```
1: i = 0
2: h# koliko zadetkov želimo
3: ponavljajociBajti = False
4: while i < |T| − k do
5:     iskalniNiz = T[i : (i + m)]
6:     decimalDatum = stringToDecimal(iskalniNiz)
7:     ponavljajociBajti = preveri(decimalDatum)
8:     if !ponavljajociBajti then
9:         steviloZadetkov = 0
10:        j = i + m
11:        while j < i + m + k do
12:            test = stringToDecimal(T[j : (j + m)])
13:            if test == decimalDatum then
14:                steviloZadetkov+ = 1
15:            end if
16:            j+ = m
17:            if steviloZadetkov ≥ h − 1 then
18:                print(i)
19:                j = i + m + k
20:                i+ = k − m
21:            end if
22:        end while
23:    end if
24:    i+ = m
25: end while
```

$\frac{|T|}{m}$-krat, v vsaki pa moramo preveriti $\frac{k}{m}$ podnizov. Časovna kompleksnost algoritma je tako $O(\frac{|T|k}{m^2})$.

3.2 Skripta za evalvacijo zadetkov

Algoritem za klesanje metapodatkov na podlagi časovnih žigov je generičen in kot tak ne zna ločevati med dejanskimi metapodatki konkretnega datotečnega sistema in naključnimi ujemanji, zato potrebujemo še algoritem za evalvacijo zadetkov, čigar izhod bodo dejanske lokacije metapodatkov. Jasno je, da tak algoritem ne more biti generičen, ampak je povsem odvisen od datotečnega sistema, iz katerega klešemo podatke.

3.2.1 Skripta za evalvacijo zadetkov datotečnega sistema NTFS

Zaradi lažje berljivosti bomo uvedli naslednje okrajšave:

- **SIA**: atribut NTFS-ja `$STANDARD_INFORMATION`,
- **FNA**: atribut NTFS-ja `$FILE_NAME`,
- **DA**: atribut NTFS-ja `$DATA`,
- **MFT**: tabela NTFS-ja Master File Table.

Skripta najprej izloči časovne žige, ki predstavljajo datum pred letom 1970 ali datum po letu 2100. Nato za vsak podani potencialni časovni žig poskuša ugotoviti, ali je le-ta del MFT vrstice.

To stori tako, da preveri, ali se časovni žig nahaja znotraj atributa SIA. Če prepoznamo glavo SIA, si pomagamo z njegovo dolžino, da se premaknemo do naslednjega atributa. Pri tem iščemo atribut DA. Če naslednji atribut ni DA in če je prvi bajt najdenega atributa manj kot 0x80, preberemo dolžino tega atributa in se premaknemo do naslednjega. Če pa je naslednji atribut FNA, lahko izpišemo njegove metapodatke in lokacijo njegovega prvega časovnega žiga dodamo na poseben seznam časovnih žigov, ki jih v prihodnje ne bomo več pregledovali. S tem si zagotovimo, da ne bomo izpisovali ponavljajočih zadetkov. Za uspešno branje atributa DA je potrebno najti vsaj en atribut FNA.

Zgornji postopek ponavljamo, dokler ne najdemo atributa DA, ali pa postopek prekinemo, če prepoznamo atribut, katerega prvi bajt je večji od 0x80. Postopek prekinemo tudi v primeru, da smo preiskali več bajtov kot je maksimalna dolžina potencialne MFT vrstice.

3.2.2 Skripta za evalvacijo zadetkov datotečnega sistema ext4

Skripta potrebuje naslednje vhodne podatke:

- tekstovna datoteka s potencialnimi časovnimi žigi,
- slika diska,
- naslov začetka particije,
- velikost bloka.

Skripta dodatno privzame velikost statičnih indeksnih vozlišč na 256 bajtov.

Podobno kot pri skripti za datotečni sistem NTFS, se orientiramo glede na potencialne časovne žige in v lokalni bližini iščemo posebne datotečne oznake:

- oznaka 0x04 predstavlja mapo,
- oznaka 0x08 predstavlja datoteko,
- oznaka 0x0A predstavlja simbolično povezavo.

Za vsako indeksno vozlišče, ki ga pregledamo, ločimo dve možnosti:

1. če indeksno vozlišče ne uporablja razširitev, preverimo, da so bajti na pozicijah od 36 do 39 (relativno na indeksno vozlišče) neuporabljeni,
2. če indeksno vozlišče uporablja razširitve pa preverimo, da se glava razširitve nahaja na poziciji 0xF30A (relativno na indeksno vozlišče).

Poleg tega se izvedejo tudi dodatni testi, ki nam povečajo verjetnost, da smo res našli indeksno vozlišče:

- test konsistence časovnih žigov [3],
- test ustrezne velikosti najdene datoteke glede na sektor,
- test ustrezne velikosti najdene datoteke glede na velikost diska.

Težava pri pridobivanju vseh metapodatkov iz indeksnih vozlišč na poškodovanem disku je v dejstvu, da se metapodatki deloma nahajajo tudi izven vozlišča, v direktorijskem zapisu. Tu lahko najdemo npr. ime datoteke in število indeksnega vozlišča. Brez uporabe nadblokov, tabele opisov skupin blokov in bitnih slik indeksnih vozlišč pa nimamo na voljo direktne povezave med njima. Za iskanje povezave si zato lahko pomagamo le s pozicijo indeksnega vozlišča ter s podatki v njem in v njegovi lokalni bližini.

Za reševanje tega problema si avtorji algoritma [6] pomagajo z indeksnimi vozlišči map, ki so na disku ostale nepoškodovane oziroma niso bile izbrisane. Tu lahko najdemo prave informacije o imenih datotek in številih indeksnih vozlišč. Postopek poteka tako: premaknemo se na prvi direktorijski zapis in preverimo, da so bajti na poziciji 4-6 enaki `0x0C0001`.

Postopek zahteva dva sprehoda čez celoten disk. V prvem sprehodu naredimo slovar števil potrjenih indeksnih vozlišč in imen datotek. Sestavimo tudi sinhronizacijski seznam. V njem se nahajajo podatki o prvem potrjenem indeksnem vozlišču mape za vsako skupino blokov. Tu zapišemo lokacijo in število indeksnega vozlišča. V drugem sprehodu s pomočjo zgoraj omenjenega slovarja in sinhronizacijskega seznama poskušamo povezati števila indeksnih vozlišč z ustreznim imenom datotek.

Za potencialna indeksna vozlišča lahko poskušamo pridobiti njihovo ustrezno število na dva načina. Prvi način je izračun preko formule: $e = dn + \frac{vl-dl}{256}$, pri čemer je e izračunano število indeksnega vozlišča, dn je število indeksnega vozlišča iz sinhronizacijskega seznama, ki predstavlja potrjeno mapo, vl je lokacija potrjenega indeksnega vozlišča in dl je lokacija indeksnega vozlišča iz mape, ki se nahaja na sinhronizacijskem seznamu. Število 256 predstavlja velikost indeksnega vozlišča.

Drugi način uporablja slovar in zgornji izračun. Vsakič, ko na zgornji način izračunamo število indeksnega vozlišča, posodobimo tudi vnos v slovarju: dodamo verzijo datoteke in čas ustvaritve. Če izračunano število indeksnega vozlišča od prej še ni obstajalo v slovarju, preprosto dodamo nov vnos. Vnos v slovar se po vnosu verzije datoteke in časa stvaritve ne sme več spreminjati. Predpostavimo tudi, da je kombinacija verzije datoteke in časa ustvaritve unikatna za vsak vnos. V prihodnje zato vsakič iteriramo čez obstoječe vnose in preverimo, ali že imamo vnos z dano kombinacijo verzije datoteke in časa ustvaritve. Če tak vnos najdemo, lahko izpišemo njegovo število indeksnega vozlišča in ime datoteke.

4. OPIS EKSPERIMENTA

Za eksperiment se je uporabil USB ključek, kateremu se je počistilo particijo z orodjem *dc3dd v.7.2.646*. Na sistemu Linux to dosežemo z ukazom:

```
sudo dc3dd hwipe=/dev/rdisk8s1 hash=md5
```

Avtorji [6] so sicer namesto sistema Linux uporabili sistem macOS Mojave v.10.14.

4.1 Reformatiranje NTFS z exFAT

USB ključek je bil nato formatiran v sistemu Windows 10 z NTFS. Na ključek je bilo naloženih 50 datotek; po 10 datotek za vsakega od 5 različnih tipov. Datoteke so bile poimenovane od File1 do File10 z dodano ustrezno končnico.

Nato je bil ključek formatiran z exFAT. Pri tem se je potrebno zavedati, da so delčki ali celotna MFT tabela ostali nedotaknjeni. Nazadnje je bilo na ključek, formatiran z exFAT, dodanih še 10 datotek.

Datoteke, dodane na ključek pred formatiranjem z exFAT služijo kot testni podatki, saj poznamo njihova imena in vsebino. Forenzično sliko diska pred formatiranjem z exFAT imenujemo *ntfsbase.dd*, po formatiranju z exFAT pa *ntfsexfat.dd*.

Zaradi lažje berljivosti bomo uvedli naslednje okrajšave:

- TP ("true positive"): zadetek, ki ga javi algoritem, ki hkrati predstavlja dejansko datoteko na *ntfsbase.dd*,
- FP ("false positive"): zadetek, ki ga javi algoritem, vendar le-ta ne predstavlja dejanske datoteke na *ntfsbase.dd*,
- FN ("false negative"): množica vseh datotek, ki jih algoritem ni zaznal, vendar so bile prisotne na *ntfsbase.dd*,
- RCL ("recall"): RCL $= \frac{TP}{TP+FN}$,
- PRC ("precision"): PRC $= \frac{TP}{TP+FP}$.

Podatki o FP in FN so bili izmerjeni s primerjanjem rezultatov, ki jih je javil algoritem in dejanskimi datotekami, ki so se nahajale na *ntfsbase.dd*. Na podlagi teh podatkov se je izračunalo tudi RCL in PRC.

Za poganjanje algoritma na tem disku je bila nastavljena velikost časovnih žigov na 8 bajtov, velikost drsečega okna pa na 24 bajtov. Število želenih zadetkov znotraj okna je bilo nastavljeno na najmanj 3.

4.2 Reformatiranje ext4 z NTFS

Za ext4 eksperiment se je v sistemu Linux Mint 18.2 počistilo ključek z ukazom `shred`. Nato je bil ključek formatiran z ext4. Nanj je bilo naloženih 25000 datotek; 500 datotek v vsakega od 50 direktorijev. Datoteke so bile poimenovane od 1.txt do 25000.txt, pri čemer število v imenu datoteke ustreza številu znakov "a" v datoteki. Razlog za izbor tekstovnih datotek je večja težavnost klesanja le-teh s strani klasičnih orodij, saj nimajo podpisa. Forenzično sliko diska v tem stanju imenujemo *expext4.dd*.

Nato je bil disk naložen v sistem Windows 10, kjer je bil formatiran z NTFS. Pri tem se je uporabila velikost gruče 4096 bajtov. Nato se je nanj naložilo 10 datotek. Forenzično sliko diska v tem stanju imenujemo *ext4NowNTFS.dd*.

Za poganjanje algoritma na tem disku je bila nastavljena velikost časovnih žigov na 4 bajte, velikost drsečega okna pa na 12 bajtov. Število želenih zadetkov znotraj okna je bila nastavljeno na najmanj 2.

4.3 Omejitve eksperimenta

- V realnem scenariju zgodovina forenzične slike ni znana in ne vemo, kateri datotečni sistem se je nahajal na njej v preteklosti.
- Verodostojnost končnega rezultata algoritma mora oceniti strokovnjak na področju digitalne forenzike.
- Algoritem ne deluje na datotečnem sistemu, kjer so bili odstranjeni ali dodani sektorji ali gruče.
- Algoritem ne upošteva datotek, na katere se nanaša več MFT vrstic, če atributa DA na najde v prvi izmed teh MFT vrstic.
- Skripta za evalvacijo zadetkov na datotečnem sistemu ext4 deluje le za primere, kjer je velikost indeksnih vozlišč enaka 256.
- Trenutna implementacija algoritma išče zadetke le na večkratnike velikosti časovnega žiga.
- Za delo z algoritmom na katerem koli novem datotečnem sistemu je potrebna izdelava nove skripte, prilagojene na ta specifičen sistem.
- Generičnost predstavljenega algoritma je le v dejstvu, da z iskanjem identičnih časovnih žigov, ki so locirano relativno blizu drug drugemu, lahko prepoznamo potencialne lokacije metapodatkov datotek. Skripte za evalvacijo teh podatkov, ki so nujno potrebne za filtriranje rezultatov prvega dela algoritma, so ne generične in težavne za implementacijo.
- Delovanje skript za evalvacijo je odvisno od proizvajalcev operacijskih sistemov - skripta, ki deluje na določeni verziji OS, morda ne bo delovala na drugi.
- Eksperimenti so bili izvedeni na zelo majhni količini podatkov.

5. REZULTATI

5.1 Pregled izvirnih rezultatov

5.1.1 Čas izvajanja algoritma

Ugotovitve o času izvajanja algoritma so sledeče:

- izvajanje prvega dela algoritma je poteklo 13 sekund,
- skripta za evalvacijo zadetkov na datotečnem sistemu ext4 je tekla 8 sekund,
- skripta za evalvacijo zadetkov na datotečnem sistemu NTFS je tekla manj kot 1 sekundo.

5.1.2 Klesanje NTFS metapodatkov

Podatki o PRC in RCL so vidni v tabeli 1. Vemo, da ima vsaka MFT vrstica en atribut SIA. Forenzična slika *ntfs-base.dd* ima 79 datotek (29 datotek in direktorijev je bilo avtomatsko generiranih s strani operacijskega sistema), torej tudi 79 atributov SIA. Dodatnih 79 atributov SIA se nahaja v $LogFile in še 4 atributi SIA v $MFTMirr. Skupaj je to 162 atributov SIA. PRC in RCL so avtorji [6] izračunali takole:

$$\text{PRC} = \frac{TP}{TP + FP} = \frac{162}{163} = 0.99 \quad (1)$$

$$\text{RCL} = \frac{TP}{TP + FN} = \frac{162}{162 + 0} = 1 \quad (2)$$

Tabela 1: PRC in RCL zadetkov v ntfsexfat.dd

	TP	FP	FN	PRC	RCL
zadetki SIA	162	1	0	0.99	1

5.1.3 Klesanje ext4 metapodatkov

Reformatiranje ext4 diska z NTFS je izbrisalo približno 20257 indeksnih vozlišč iz tabele indeksnih vozlišč. Od indeksnih vozlišč, ki so ostala, pa je algoritem našel vse [6].

Da bi lahko izmerili RCL in PRC, so avtorji ustvarili še eno kopijo ext4 forenzične slike in vsem 25000 tekstovnim datotekam ter 50 direktorijem dodali še število indeksnega vozlišča kot atribut. To so storili z Linux ukazom `attr`. To forenzično sliko diska imenujejo *expext4Attr.dd*. Po formatiranju z NTFS pa dobijo še eno forenzično sliko, ki jo imenujejo *ext4AttrNowNTFS.dd*. S pomočjo teh dveh kopij so primerjali izračunane števila indeksnih vozlišč s pravimi.

Avtorji so izvedli dve meritvi. Prva meritev meri, koliko izračunanih števil indeksnih vozlišč dejansko ustreza pravim. Rezultati se nahajajo v tabeli 2 in tabeli 3. Druga meritev meri, koliko je algoritem sposoben prepoznavati indeksna vozlišča, ne glede na to, ali pripadajo testnim datotekam in direktorijem ali ne. V tem primeru za FP vzamejo vse zadetke, ki v resnici niso indeksno vozlišče. Rezultati so vidni v tabeli 4 in tabeli 5.

Tabela 2: PRC in RCL zadetkov v expext4Attr.dd

	TP	FP	PRC	RCL
najdeno št. ind. vzl.	77481	0	1	1
izračunano št. ind. vzl.	27336	50145	0.35	1

Tabela 3: PRC in RCL zadetkov v expext4AttrNowNTFS.dd

	TP	FP	PRC	Najdene datoteke
najdeno št. ind. vzl.	15544	41692	0.27	4848
izračunano št. ind. vzl.	7091	50145	0.12	5755

Tabela 4: PRC zadetkov v ne-formatirani forenzični sliki

TP	FP	PRC
77675	0	1

Tabela 5: PRC zadetkov v formatirani forenzični sliki

TP	FP	PRC
57427	0	1

6. PONOVITEV EKSPERIMENTA

6.1 Okolje izvajanja

Pri ponovitvi eksperimenta smo uporabljali 16 GB USB ključek, virtualno okolje z operacijskim sistemom Linux Ubuntu ter operacijski sistem Windows 10. Da na rezultate ne bi vplivale naključne, prej naložene datoteke na ključu, smo pred začetkom vsakega eksperimenta USB ključ najprej prepisali z ničlami.

```
dc3dd hwipe=/dev/sdb
```

V nadaljevanju smo z uporabo njihovih orodij, ki so dostopna na platformi GitHub[1], izvedli podobna eksperimenta, kot jih izvedejo avtorji v svojem članku [6]. Naša eksperimenta sta se pri tem do neke mere razlikovala v številu in vsebini datotek, zapisanih na USB ključ, preden smo le-tega formatirali. Posledično smo dobili nekoliko drugačne rezultate, vendar smo vseeno uspešno potrdili, da so uporabljena orodja [1], ki so jih avtorji članka [6] razvili, zelo uspešna pri klesanju metapodatkov.

6.2 Reformatiranje NTFS z exFAT

USB ključ smo v sistemu Windows 10 formatirali v format *NTFS*. Nanj smo prenesli 50 datotek, kjer jih je bilo po 10 istega tipa. Pomembno je izpostaviti, da smo datoteke naključno izbrali z diska enega izmed članov naše skupine. To pomeni, da je zelo velika verjetnost in lahko upravičeno pričakujemo, da vsi časovni žigi neke datoteke ne bodo enaki in se bo čas zadnje spremembe razlikoval od časa kreiranja datoteke.

USB smo nato priključili v Linux ter najprej ustvari kopijo diska poimenovano *ntfsbase.dd*. Na njej smo kasneje izvedli klesanje metapodatkov, rezultate pa smo uporabili kot osnovo pri primerjavi s formatiranim diskom.

```
dc3dd if=/dev/sdX of=/path/ntfsbase.dd
```

Na USB napravi smo nato ustvarili datotečni sistem *exFAT* in na napravo dodali 10 praznih tekstovnih datotek. Takoj za tem smo naredili sliko diska, poimenovano *ntfsexfat.dd*, ki nam je služila za forenzično preiskavo.

```
dc3dd if=/dev/sdX of=/path/ntfsexfat.dd
```

Sliki *ntfsbase.dd* ter *ntfsexfat.dd* smo analizirali z orodjem *cPTS*.

```
cPTS ntfsbase.dd 8 24 3
cPTS ntfsexfat.dd 8 24 3
```

Dolžino časovnega žiga smo nastavili na 8 bajtov, dolžino preiskovalnega okna 24 bajtov ter število ujemanj na 3. Rezultata skripte smo shranili v datoteki *cPTS_ntfsbase.txt* ter *cPTS_ntfsexfat.txt*. V zadnjem koraku smo dobljene potencialne časovne žige obeh slik podali kot vhod skripti za evalvacijo zadetkov.

```
ntfsParser.py cPTS_ntfsbase.txt ntfsbase.dd
ntfsParser.py cPTS_ntfsexfat.txt ntfsexfat.dd
```

6.3 Reformatiranje ext4 z NTFS

Postopek drugega eksperimenta je bil zelo podoben prvemu, zato bomo opisali zgolj ključne korake.

Na USB ključu smo najprej ustvarili datotečni sistem ext4. Ustvarili smo dva imenika ter v vsakem izmed njih po 5 tekstovnih datotek, ki smo jih oštevilčili od 1 do 10. V vsako datoteko smo zapisali x ponovitev niza “aaaaaaaaaa” (10 krat znak ‘a’), kjer je x številka te tekstovne datoteke.

Ta oblika diska nam bo služila kot osnova, zato smo shranili njeno sliko.

```
dc3dd if=/dev/sdX of=/path/expext4.dd
```

USB ključ smo nato priključili v sistem Windows 10 ter napravo formatirali. Ustvarili smo datotečni sistem NTFS, kjer smo uporabili velikost gruče 4096 bajtov. Na ključu smo ustvarili 10 novih, praznih, tekstovnih datotek. USB ključ smo priključili nazaj v sistem Linux in ustvarili njegovo kopijo.

```
dc3dd if=/dev/sdX of=/path/ext4NowNTFS.dd
```

Ponovno smo na obeh slikah najprej izvedli program *cPTS*, za tem pa potencialne časovne žige analizirali z ustrezno skripto za evalvacijo zadetkov.

6.4 Rezultati

6.4.1 Reformatiranje NTFS z exFAT

Na obeh slikah diskov je orodje našlo večino zapisov. Našlo je SIA, FNA ter DA zapis za vsa *posebna* polja tabele MFT, ne pa tudi za zapise datotek. Orodje je našlo in izpisalo 49 od 50 zapisov FNA za datoteke, ki smo jih naložili na NTFS disk. Datoteke “File7.gpx” ni našlo ne na reformatirani sliki, kot tudi ne na osnovni sliki.

Zanimivo je, da je orodje izpisalo zgolj FNA zapise, ne pa tudi zapisov SIA. Kot avtorji v članku navajajo, so uspešno našli SIA zapise za vse datoteke. Naši rezultati se zelo verjetno razlikujejo od rezultatov iz članka, ker smo uporabili neke že obstoječe datoteke, ki niso imele enakih vseh štirih časovnih žigov, ki jih datotečni sistem hrani za posamezno datoteko. Podrobneje, ta razlika verjetno izhaja iz dejstva, da se v zapis FNA shranijo časovni žigi trenutka, ko je datoteka ustvarjena na neki particiji, medtem ko se nekateri časovni žigi v zapisu SIA ohranijo [4]. Namreč ravno primerjava SIA in FNA zapisa neke datoteke nam v splošnem

omogoča da opazimo, ali je bila datoteka kako spremenjena oziroma prenešena na drug medij.

Zaključimo lahko, da je bilo orodje relativno uspešno, saj je za 98 odstotkov naših datotek našlo FNA zapis in pravilno izpisalo vse metapodatke, ki jih le-ta vsebuje. Vseeno je ena izmed možnih izboljšav orodja ta, da bi orodje iskalo tudi SIA in DA zapise glede na FNA zapis, saj lahko pričakujemo, da bomo več enakih časovnih žigov našli ravno v FNA zapisu in ne SIA.

6.4.2 Reformatiranje ext4 z NTFS

Na obeh slikah diskov je program uspešno izpisal metapodatke za vseh 10 tekstovnih datotek, ki smo ji zapisali na disk z datotečnim sistemom ext4, preden smo le-tega formatirali na NTFS. Za katero datoteko gre lahko ugotovimo glede na znano velikost tekstovnih datotek.

Program upešno izpiše podatke, ki so shranjeni v indeksnem vozlišču. Izpiše tudi informacije o glavi obsega podatkov (ang. "extent header information") ter tudi same podatke o lokaciji podatkov, torej fizično lokacijo bloka, logično številko bloka ter število blokov.

Orodje se je torej izkazalo kot zelo uspešno, saj smo uspešno pridobili metapodatke o vseh desetih datotekah. Pri pregledovanju rezultatov smo opazili, da na vsako datoteko oziroma na fizični blok, kjer je neka datoteka zapisana, kaže več indeksnih vozlišč. To je edina stvar, ki je bila v tem eksperimentu proti pričakovanjem, saj na disku nismo ustvarjali bližnjic ali povezav in smo zato za eno datoteko pričakovali natanko eno indeksno vozlišče.

7. ZAKLJUČEK

Ogledali smo si inovativen način klesanja metapodatkov, ki temelji na časovnih žigih [6]. Kot so že raziskali avtorji članka [6], datotečni sistemi na zaporednih naslovih hranijo časovne žige, katerih vrednosti se pogosto nekajkrat ponovijo. Z našimi eksperimenti smo potrdili ugotovitve avtorjev, da je iskanje metapodatkov na podlagi enakih časovnih žigov do neke mere lahko smiselno in učinkovito. Tako, kot v članku [6], smo tudi sami prišli do praktično popolne natančnosti orodja, ko smo iskali metapodatke novo ustvarjenih datotek. Poudariti je potrebno, da smo tako kot že avtorji sami, imeli precej težav pri klesanju metapodatkov iz diska, ki je vseboval naključne datoteke. Uporabniške datoteke se običajno večkrat spreminjajo in prenašajo na različne medije, zato se pri takem tipu datotek časovni žigi navadno razlikujejo. Tako je bilo na primer v našem eksperimentu orodje 0% uspešno pri pridobivanju SIA zapisov. Ker je šlo slučajno za datotečni sistem NTFS, smo del metapodatkov, zapis FNA vseeno uspešno pridobili za vse datoteke. Vseeno se moramo zavedati, da če bi šlo za nek drug datotečni sistem, verjetno ne bi uspeli pridobiti iskanih metapodatkov. Zaradi te omejitve smo že predlagali izboljšavo programa *NTFS_parser*, v splošnem pa se trenutno izkaže, da pri trenutni implementaciji orodja *cPTS* lahko pričakujemo večjo uspešnost klesanja metapodatkov sistemskih datotek, kjer je večja verjetnost, da se vsaj dva časovna žiga neke datoteke ujemata. Možna nadgradnja orodja *cPTS* je tudi, da bi pri iskanju "enakih" zaporednih časovnih žigov omogočili poljubno časovno razliko, ki bi jo lahko uporabnik orodja izbral sam. S tem bi zelo verjetno našli več datotek, hkrati pa se postavi vprašanje, koliko bi zvečali število lažno-pozitivnih zadetkov in ali je potem to razmerje med TP ter FP še vedno sprejemljivo.

8. VIRI

[1] cpts. `https://github.com/RuneN007/cPTS`. [Dostopano 26.5.2021].

[2] ext4 disk layout — Wiki. `https://ext4.wiki.kernel.org/index.php/Ext4_Disk_Layout`. [Dostopano 15.3.2021].

[3] A. Dewald and S. Seufert. Afeic: Advanced forensic ext4 inode carving. *Digital Investigation*, 20:S83–S91, 2017.

[4] J. Fichera and S. Bolt. Chapter 6 - host analysis. In J. Fichera and S. Bolt, editors, *Network Intrusion Analysis*, pages 153–167. Syngress, Boston, 2013.

[5] L. Naiqi, W. Zhongshan, H. Yujie, and QinKe. Computer forensics research and implementation based on ntfs file system. In *2008 ISECS International Colloquium on Computing, Communication, Control, and Management*, volume 1, pages 519–523, 2008.

[6] R. Nordvik, K. Porter, F. Toolan, S. Axelsson, and K. Franke. Generic metadata time carving. *Forensic Science International: Digital Investigation*, 33:301005, 2020.

[7] A. Pal and N. Memon. The evolution of file carving. *IEEE Signal Processing Magazine*, 26(2):59–71, 2009.

[8] R. Poisel and S. Tjoa. A comprehensive literature review of file carving. In *2013 International Conference on Availability, Reliability and Security*, pages 475–484, 2013.

[9] R. Russon. NTFS documentation — NTFS-Linux project. `https://flatcap.org/linux-ntfs/ntfs/attributes/index.html`. [Dostopano 15.3.2021].

Searching for the source of images based on information in image headers

A simple machine-learning approach

Jan Pavlin
Faculty of Computer and Information Science
Večna pot 113
Ljubljana, Slovenia
jp8765@student.uni-lj.si

Moritz Voigtländer*
Faculty of Computer and Information Science
Večna pot 113
Ljubljana, Slovenija
mv5741@student.uni-lj.si

ABSTRACT

While there are many different approaches to trace the provenance of an image many of them are demanding in resources and complex to build, which makes them costly to apply to individual cases of digital forensic. We investigate an approach of grouping JPEG images by their file headers, which is already considered as lightweight, and try to simplify it even more to make applying a machine-learning model feasible in everyday digital forensic investigations while still reaching applicable results. We refer to the ideas of [1] P. Mullan et. al. (2020), stripping their models from even more complexity unifying it to one model. Our findings (overall accuracy 98%) support that the path of building lightweight machine-learning models for grouping images by provenance is actually promising as suggested in the original paper.

1. INTRODUCTION

In digital forensic investigations data needs to be grouped, to provide a good overview and leading to possible insights where data answering questions on the case can be found. The forming of groups needs to support the specific case, but it is usual to group by same datatype and the source of the evidence, e.g. images and their provenance. For further analysis digital investigators aim to trace images back and group them by their sources. Methods to reach this goal include exploiting pixel characteristics, or gaining metadata from the image. While dealing with specific pixel characteristics is very costly in terms of building models and computational effort, metadata is known to be less reliable due to its openness to manipulation but suitable for lightweight approaches dealing with a huge amount of images, which is often the case in digital forensic investigations.

*ERASMUS exchange student from IFI, LMU Munich

In this essay we focus on data already limited to JPEG-images but from different sources - here source must not mean the location an image is found at, but the device used creating the image. From the JPEG-images we extract the header information consisting of EXIF-tags. Image metadata is prone to manipulation especially as they are normally stored in standardized formats such as e.g. EXIF, IPTC or XMP, where many programs are available for changing the metadata and therefor manipulating header information of images, making it feasible also for users with average knowledge. In a digital forensic investigation we need to take the technological knowledge of the suspect into account, whether it is likely that the suspect uses applied knowledge to manipulate data leading investigators into wrong directions, or actually hiding evidence. While in cases of long planned action the suspect probably will try to influence as much evidence as possible in its favor (destroying traces or manipulating), at least in accidental happenings, acts in the heat of the moment and in many cases of suspects lacking the knowledge of generated digital evidence, the header information will not have been tampered with on purpose. Yet there is the possibility that images have been post-processed using software which sometimes change header information - desktop software being more likely to change it than smartphone apps. [1]

Using information from the header files of images about the camera model, make or software for post-processing are a promising source for grouping images by provenance, since images produced from the same source likely share in parts the same header information. While the format of image headers is standardized the actual use and content of individual fields in the header is not - leading to various different usage-schemes across camera-makes. Yet intra-make the models of one make share large parts of this makers usage-scheme allowing for finding patterns to identify the source make of an image regardless if the specific model is already known.

2. RELATED WORKS

Several methods have already been developed to identify devices that have captured images or the software with which the images have been processed. McKay [2] proposes a model that separates camera, scanner, and computer-generated images, as well as a device model. For these predictions it

uses the support vector machine on the data obtained from the image. Classification of the camera model is conducted in many other works. For example, Bayar [3] and Bondi [4] use neural networks.

While some researches try to extract features from the image, others choose a slightly simpler solution and build features from the image headers [5, 6]. The advantage of such an approach is that not the entire image needs to be processed, which leads to a very lightweight solution that can learn on many images. However, using image headers can lead to additional challenges. Additional processing or different social networks greatly change the values of the data stored inside image headers.

Another problem in the real world arises from the constant development of new models of phones and cameras. Models built on older data sets do not qualify for the classification of data on which they did not learn. Because of this problem we need to prepare a solution that will also be able to solve an open-set problem, where it will encounter data that it has not yet seen. We found only few solutions to this problem; [4] introduces a model that identifies unknown camera models and [7] tackles the open-set problem of deep learning.

3. CLASSIFICATION OF MAKES OF UNSEEN MODELS

We implement a procedure that - as in the original article [1] - is able to classify images to their correct source camera-make while the individual capturing device is a previously unseen camera model from that make. The code is available at:
https://github.com/JanPavlin/DataForensics.

3.1 Data acquisition and preparation:

Authors of [1] P. Mullan et. al. (2020) provided a *.csv* file, which contains web links to almost 3 million flickr-images, as well as information about camera makes, models and software. Based on the data in these files we select and filter the nine most common camera makes. Those are Apple, Canon, Fujifilm, Kodak, Nikon, Olympus, Panasonic, Samsung and Sony. As suggested in the original paper we are using only models for which a minimum of 200 images are present in the dataset. To have a diverse set for each camera make 20 random models are selected and for each model we sample 200 different images. Using the flickr API, we download all the selected images but saved only their header information. As a result our dataset consists of 34663 image headers.

3.2 Machine learning:

From the obtained data we built a second dataframe, where each row represents one unique downloaded image, and attributes represent all possible EXIF tags from the images. The value of a specific cell equals the value of the image tag if the tag is given for this image, or no value if the tag is missing for this image. The dataframe has 34663 rows (number of images) and 204 columns (number of possible distinctive EXIF tags). We find 204 different EXIF tags are being used and calculate, that an average image uses only 44 tags of that 204. That means, 78.5% of values within this dataset are value $None$.

The key idea is now to transform this dataframe in a way, that instead of the value of a specific cell, a True/False Boolean is used to indicate whether the data of a specific tag was present for a given image or not. So, each field that had data is assigned a value of 1 (True) and the rest a value of 0 (False).

The idea behind such a format is that it does not matter what the GPS coordinates or creation time of the image are, but whether the camera that captured the image also recorded that tag. We assume that information whether the tag is recorded at all is more important than the value of that tag itself, when it comes to finding a pattern, which helps to predict the make of an unknown model.

The machine learning is done starting with the following steps:

- At start, one camera model is randomly selected. This model is removed from the main dataset and will serve as a test set of images from an unknown model. Since each model contains 200 images and we have 180 models ($9 camera makes * 20 models$) only one model represents a really small percentage of the data.
- With the use of python´s scikit-learn library [8] we build our classifier. Since in the original paper random forest classifier is used successfully, we decided to do the same. Random forest classifier is an ensemble learning method that operates by constructing multiple decision trees when training and predicting the class that is the mode of the classes of the individual trees [9]. The model is build on a training set that consisted of 179 models with 200 images each.
- When the learning was finished, the ML-model is used to predict the $camera\_make$ of the test images. For each iteration we calculate the accuracy, f-score, precision and recall.

3.3 Results:

The experiment is repeated 50 times. To evaluate our results we calculate the average accuracy of the model and then generate the confusion matrix.

In the confusion matrix (Figure 1) we can see, that our model performs extremely good for all $camera\_makes$. Over 50 iterations, 12692 images are included as test set and on average, our model achieves an accuracy of 98%. The only exception is Fujifilm camera-make, with an accuracy of 78.8%. When failing, the model usually predicts Nikon (16.5%) and Apple (4.5%). Highest accuracy is achieved on Kodak camera makes. Here, the model fails only one time in 2120 images.

Interestingly, the model can predict $'camera\_make'$ with very high accuracy. We check the feature importance of the classifier itself, as well as the calculated feature importance using MDI (mean and standard deviation).

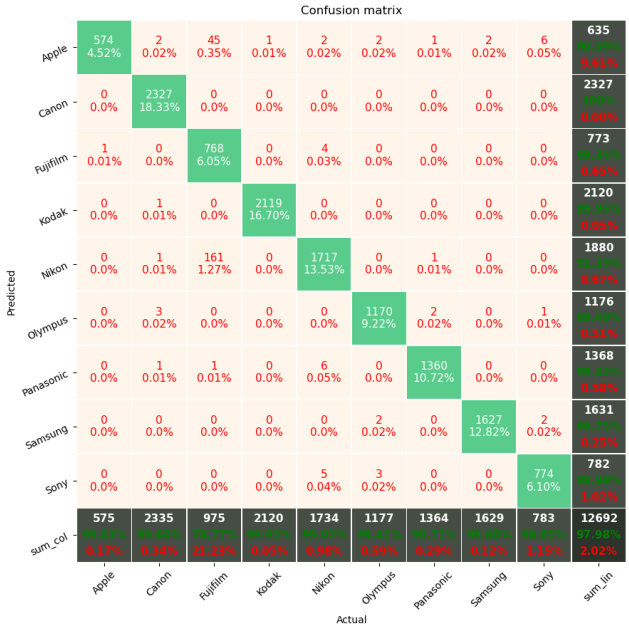


Figure 1: Confusion matrix [10]

The most important EXIF tags and their coefficients are:

- 37379 (BrightnessValue): 0.1453
- 41493 (ExposureIndex): 0.1410
- 41996 (SubjectDistanceRange): 0.1267
- 41495 (SensingMethod): 0.1247
- 41985 (CustomRendered): 0.1078
- 41991 (GainControl): 0.1045

For a clearer display, for each EXIF tag we build graphs of missing values of every $'camera\_make'$ types, which allows to understand differences in input-schemes across makes. This is shown in Figure 2.

We observe that on each of these graphs the group of models reaches 100% of the missing values, while the others have very few or almost no missing values. Let´s say that we receive an image that has data for tag 37379 (BrightnessValue), tag 41996 (SubjectDistanceRange) and 41495 (SensingMethod), but tag 41493 (ExposureIndex) is empty. Based on graphs on Figure 2 we can be fairly certain that image was captured with a Fujifilm camera.

4. CONCLUSION

In this essay we focus on searching the source of images based on information in the image headers following the idea of [1] P. Mullan et. al. (2020), but using a simpler approach. As they state already in the original paper, header information is very fast and easy to collect and allows for a lightweight solution to grouping large amounts of data - yet is on one hand prone to manipulation and on the other hand source identification is an inherent open-set problem adding potential new sources with every new device being developed. We present our approach to classification of camera makes of unseen models. We use flickr API to download the images and retrieve their header information. A random forest classifier is trained on all but one camera models and tested on images of the excluded model. Our simplification consists of just controlling for the presence of EXIF-tags in the header information, and not dealing with the actual values. Our work suggests it is of high potential to predict the make of an image created with a possibly unknown model by just building a pattern around the scheme of EXIF-tag-usage.

In the future it would be interesting to apply the model in digital forensic investigations and to find out how the model performs when it is combined with other, more computationally demanding models.

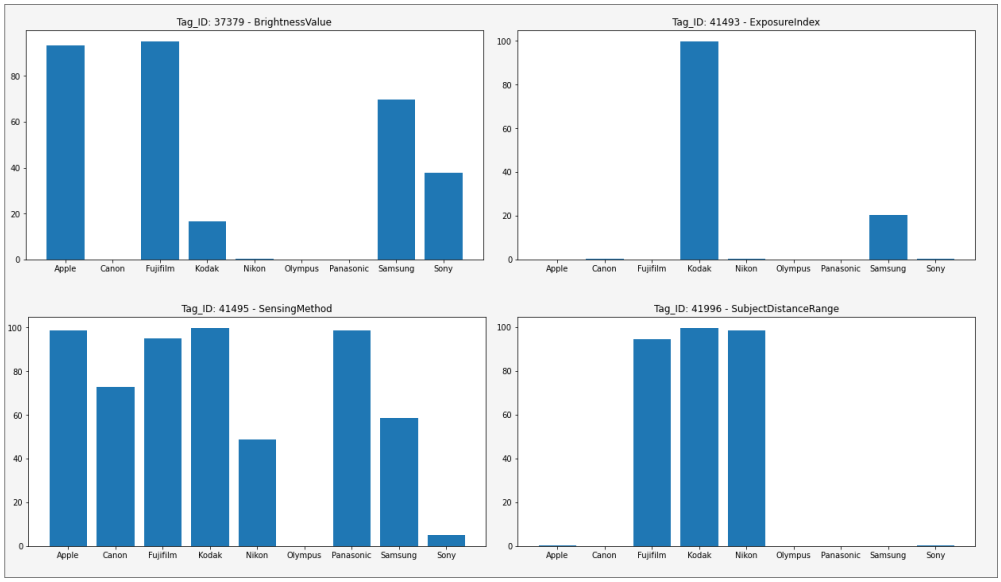


Figure 2: Percentage of not missing values for different EXIF tags across makes

5. REFERENCES

[1] Mullan P., Riess C., and Freiling F. Towards open-set forensic source grouping on *JPEG* header information. *Forensic Science International: Digital Investigation*, 32:300916, 2020.

[2] Christine McKay, Ashwin Swaminathan, Hongmei Gou, and Min Wu. Image acquisition forensics: Forensic analysis to identify imaging source. In *2008 IEEE international conference on acoustics, speech and signal processing*, pages 1657–1660. IEEE, 2008.

[3] Belhassen Bayar and Matthew C Stamm. Augmented convolutional feature maps for robust cnn-based camera model identification. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 4098–4102. IEEE, 2017.

[4] Luca Bondi, Luca Baroffio, David Güera, Paolo Bestagini, Edward J Delp, and Stefano Tubaro. First steps toward camera model identification with convolutional neural networks. *IEEE Signal Processing Letters*, 24(3):259–263, 2016.

[5] Patrick Mullan, Christian Riess, and Felix Freiling. Forensic source identification using jpeg image headers: The case of smartphones. *Digital Investigation*, 28:S68–S76, 2019.

[6] Eric Kee, Micah K Johnson, and Hany Farid. Digital image authentication from jpeg headers. *IEEE transactions on information forensics and security*, 6(3):1066–1075, 2011.

[7] Belhassen Bayar and Matthew C Stamm. Towards open set camera model identification using a deep learning framework. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 2007–2011. IEEE, 2018.

[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[9] Tin Kam Ho. Random decision forests. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, volume 1, pages 278–282 vol.1, 1995.

[10] Wagner Cipriano. Pretty print confusion matrix. https://github.com/wcipriano/pretty-print-confusion-matrix, 2020.

IoT Botnet Forensics

A Comprehensive Digital Forensic Case Study on Mirai Botnet Servers

Andraž Juvan
aj6202@student.uni-lj.si

ABSTRACT

Botnets are a relatively new thing in the world of IoT. Since the attack vectors now range from smart refrigerators to security cameras instead of just different types of operating systems we don't yet know the best approaches to tackle these infections forensically. Despite the low performance of these kinds of devices they are widely spread and can cause serious outages due to their numbers. Additionally we learn to protect our computers using antivirus software, installing updates, etc. we usually never do that with IoT devices which decreases the chance of mitigating a botnet's spread.

In the year 2016 the Mirai botnet was activated and it took down more than a 1000 server which included widely used sites like Twitter, Netflix and DynDNS as shown in figure 1. And with the release of its source code, even more botnets were created which started using these devices for different malicious purposes. This study is focused on this exact botnet and its variants that came from the released source code to help the forensic researcher's to discover, monitor and act on active new implementations. The components of the botnet, like the Command and Control (CNC) server, Scanner and Loader, are described in detail from a forensic point of view. The descriptions contain different data collection methods, parsing the information from the data and breaking down the core parts of the code and network traffic which can be used for obtaining useful information about new similar botnets and their mitigation.

1. INTRODUCTION

A botnet is a group of various internet enabled devices that have been compromised by an attacker. They tend to multiply exponentially on their own by exploiting a vast list of vulnerabilities leaving a trail of dormant code that the mentioned attacker could activate at any time. With the age of Internet of Things (IoT for short), where every home appliance is connected, a carefully designed botnet could spread uncontrollably. And because we don't actively update and patch our IoT refrigerators like we do with our computers, the attackers can take advantage of this large attack vector to practically no limit.

The first IoT botnet attack of this size happened in the year 2016, when the so called **Mirai** botnet shut down a large portion of the internet. This ranged from the biggest known websites, like Twitter, Netflix and DynDNS (shown in figure 1), as well as a few major Russian banks and practically the entire country of Liberia. Later in the year 2017, the source code for the Mirai botnet was released to the public, which has lead to other botnets being developed. The different variants that were developed mostly just had more or newer exploits, attacked other devices as well as IoT devices, acted as spammers and in some cases even as crypto miners.

Investigating botnets poses a challenge, since it takes different approaches than the classical digital forensics. The first challenge is collecting the executable code. That can be obtained by analysing an infected device or collecting the information via a honeypot. From there the forensic scientist can analyse the code and figure out what kinds of exploits it's using to spread, what method it's using to activate, if there's a potential kill switch which can be used to mitigate the spread of the attack, etc. But the issue is that only using the information about the botnet executable is not enough. The general approach for creating botnets, which is the case with Mirai, is to have a control node for activating it and the important information is stored there. If the researchers were to get their hands on this machine, they could work their way through the initial infections to whole botnet.

Since Mirai was the first IoT attack with such a large impact, forensic scientists were trying to answer the questions, what kind of approaches and techniques could be used on botnets and what kinds of evidence, and with it investigative information, could be obtained. The described paper is the first forensic case study for these types of attacks. As a part of this study the following assets were available: disk image, memory (RAM) image, and network traffic from the Mirai's control servers. The goal of the study is to find the list of infected IoT devices, the record of executed DDoS attacks, recovery of login credentials and exporting a list of infected devices.

2. RELATED WORK

Since the Mirai attack has been executed a good number of papers were written on the topic of botnets and more specifically the Mirai botnet. They describe how the attack

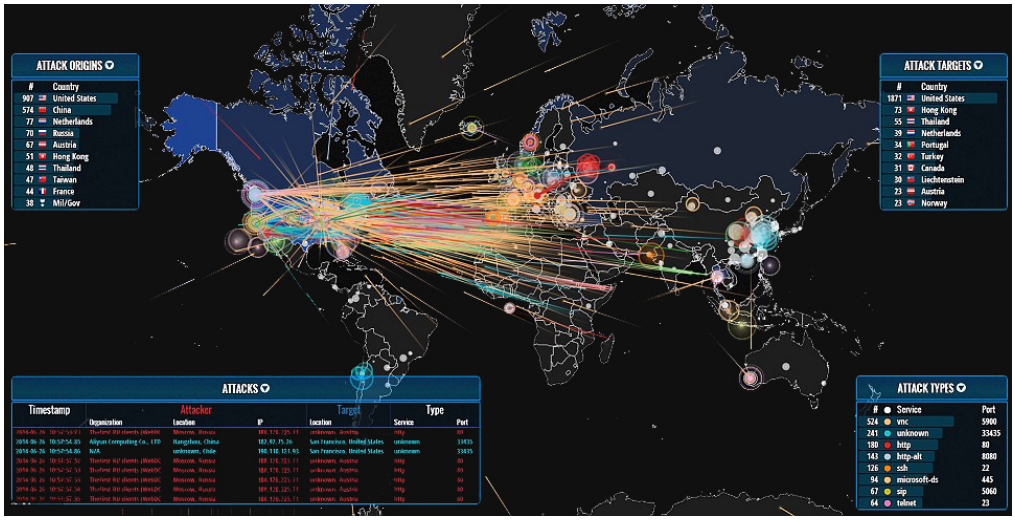


Figure 1: Norse live attack map showing the attack on May 2016 targeted towards DynDNS.

works, what is needed to set it up, the analysis of forensic artifacts, and more. The resulting papers prepared the grounds for a new approach of digital forensics in the world of botnets. While this botnet is currently the most known IoT botnet, because of its large attack vector, it's not the first of its kind. These types of botnets age back since the days when IoT technology just started blooming. In the year 2013 researchers discovered a worm called **Linux.Darrloz** [1] which exploited a PHP vulnerability in home IoT devices like routers, TV boxes, security cameras, printers, etc. Since then a number of other worms and botnets were found that utilized different vulnerabilities and had different intents.

With new and improved versions of botnets on the horizon, it's important to note what can be done to protect the users and prematurely notice and act on the spread of a new botnet. In the following years, after the attack, a number of botnets were found which were branches of the original Mirai botnet. Their functionality was mostly the same, but some were also altered to include more vulnerabilities, spam users and in some cases even for crypto mining purposes. Some of the more notable branches were:

- **IoTroop**, which also attacked different types of routers and allowed for patching the active botnets.
- **Echobot**, which included the exploits of a couple enterprise applications, like Oracle WebLogic and VMWare SD-WAN.
- **Emotet**, which attacked the victim's mailboxes and in some cases even stole credentials.
- **Gamut**, which was used for spamming victims' emails.
- **Necurs**, which acted as a ransomware.

2.1 Overview

After the publication of the Mirai source code along with the documentation, the following key items were found to be needed:

- **Command and Control (CNC) server** the acts as the main server which communicates with the rest of the botnet, coordinates attacks, counts the infected devices, etc.
- **MySQL server** for storing the information used by the CNC server.
- **Scan receiver** which acts as an endpoint for collecting the vulnerable device's information and forwards it to the loader.
- **Loader server** used to log in to the vulnerable device, acquired from the scan receiver, and install the malware.
- **DNS server** that stores all the information about the servers, so the attacker is able to change their locations when needed.

As shown in figure 2 the attacker can launch the attack by sending a command from a remote terminal to the CNC server by using telnet. After that the server saves the attack information to MySQL and is at the same time transmitted to the infected IoT devices. The running devices then start flooding the targeted server(s) with network packets and try to take it offline.

But before the attack takes place the attacker needs to spread the botnet enough for the attack to succeed. This process starts with the initially infected device which finds a few other devices that can be accessed. Those are then attacked and exponentially start infecting other devices in their network and additionally exploring the wider web for new publicly available and vulnerable devices. This is done by first finding the vulnerable service, like a badly configured telnet or SSH, and their credentials. Those are then forwarded to the Scan receiver, which logs the devices and instructs the Loader to try and exploit them. It's important to note that the connection between the Scan receiver and Loader and made through the CNC server with the server

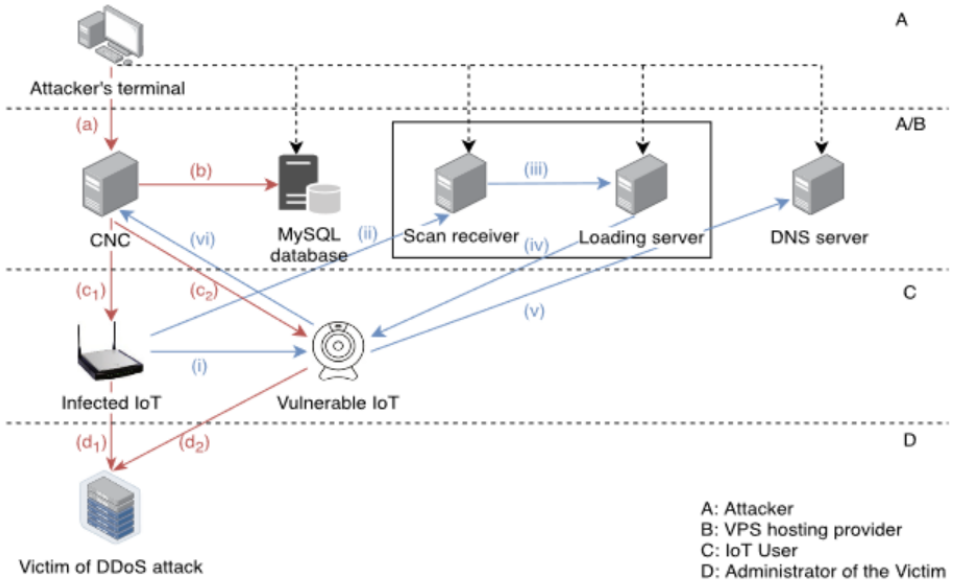


Figure 2: Mirai botnet topology [3] described by the GitHub document and set up by the researchers during their investigation.

constantly checking for information on the mentioned machines. That way the machines could be recycled and it'd be harder to track the information back to the CNC server, even more so since the CNC server was connected through the Tor network [2]. After the Loader installs the malware on the vulnerable device they can then continue to spread. It's important to note *how* the Scan receiver and the CNC are contacted here. Like shown above the infrastructure uses a DNS server to contact the needed services, which means that the attacker can change the location (aka. the IP address) of the service, but also that by removing the DNS entry, or rather by blocking the specific entry, you could mitigate the attack on a network.

2.2 Methodology and environment

To analyse how the botnet works we need to set up the above mentioned environment and note what we want to analyse or rather what information we want to gain from looking into it's inner workings. The key points of the analysis are best brought up by the described paper [3]:

- Setting up the environment using the publicly available code from GitHub and the exact network topology as shown in figure 2.
- Acquire the data from the file system, memory and network from each physical server.
- Manually analyse the acquired data to try and recover the history of the attacks, the victims of the attacks and the infected devices.
- Verify the findings with Mirai's source code which were then incorporated into a road map made for these types of attacks.

3. ARTICLE

Most other articles described the attack, wrote the details about the environment, noted concerns, etc. while the described article [3] on top of that also focused on how forensic scientists should approach botnets, what kind of information we can gather from them, and described details about the inner workings of the botnet. Before analysing forensic artifacts we must first define what they are and what we want from them. The components we want to get our hands on are the Attacker's terminal, CNC server, database server, scanner, loader and the DNS server. From them we want to extract the history of commands, network traffic, estimated botnet size, etc. along with reverse engineering the executables to see it's inner workings.

3.1 Attacker's terminal

The first component we'll take a look at is the attacker's terminal, which the attacker uses to communicate with the CNC server. To connect to the server the attacker first establishes a telnet connection and passes valid credentials. Upon successfully connecting to it the attacker can then execute commands via a dedicated shell.

From the forensic scientist's perspective this is a huge plus, since the telnet protocol does not use any kind of encryption. This means that they can sniff the traffic and read the login credentials as well as the issued commands. If the scientist could get their hands on the CNC's network history, they could acquire a list of commands executed with timestamps, victims and potentially number of bots that were used to attack.

3.2 CNC and database server

This server is the most important piece of software in the botnet, since every new bot is registered through it and it's

used to communicate with them when an attack is executed. That is why it's the most sought after forensic artifact in the topology, from which we can extract the executable file, memory data and network traffic.

Using the source code available to us we can see that the information for the database, like credential and location, are hard coded into the file, therefore we can extract it directly from the executable. From there we can access the database and see information like the history of commands, the users, and a whitelist table which consists of IP ranges that the bots mustn't scan.

Since the credentials were extracted above from the CNC server the forensic scientist should conduct their examination through the remote and physical access. Once accessed the standard action is always to create an image of the server before conducting the analysis. From that we could look at the database files created by MySQL and check the contents. On a remote connection that can also be done using the MySQL's dump tool and going forward from there.

Along with the static data analysis we can also take a look at how it runs and what's interesting to a forensic scientist in the executing process. Looking at the code we can see that the CNC server has three queues which contain active bots, deleted bots, and bots executing the attacks. If the memory image is available the forensic scientist can look through it to see these queues, which is in this case a vital piece of forensic data. Since the server was written in GoLang it's hard to determine the low level implementation of the queues, so we must resort again to reverse engineering it from the executable. After that the queue can be traversed and the IP addressed can be found with a few pointer hops though a few structures.

Now we could also analyse the traffic independently. The payloads were constructed by short byte codes representing the data needed to execute the attack constructed like:

- **Bytes 3 to 6**: Duration of the attack in seconds.
- **Byte 7**: Attack type which is already implemented in the binary.
- **Bytes 9 to 13**: Victim's IP and mask that represent the range of the attack.

The omitted bytes don't represent any important information at this time.

Besides the packet ordering attack execution we also have the registration and ping packets coming from the botnets themselves. The registration packets are sent when a new botnet device becomes active, so after the loader installed it. It sends the packet to the CNC server that then saves it into the database and also marks the botnet as active. After the new botnet device is registered it sends pinging packets that mark the device as active and if it stops receiving the pings it marks the device as deleted.

3.3 Scan receiver and Loader

Since all the newly discovered vulnerable devices go through these two services, we can find a lot of information by scanning the network traffic. With its examination we find that packets sent to the receiver consist of the IP, port which in turn describe the vulnerable service, and the username and password limited by their length.

The technique used to forward the data to the loader vary from implementation to implementation. In our case it's transmitted over the standard output to the loader's standard input, while some other implementations send this data over the CNC server via tor. The standard output stream is not stored in a file but rather passed over the memory and is quickly deleted, therefore the forensic scientist must follow the file socket or pipe to get the information from there. And since the transmission protocol is simple, since it consists of sending the previously mentioned data with simple delimiters, we can easily extract that information and save for later use. This way we can track the history of the infections and which services were targeted.

3.4 DNS server

The last piece of the puzzle is the DNS server. It's used to forward the IP of the critical services, like the CNC and the scan service, to the infected machines. By capturing the network we can determine which machines were infected since they used the DNS server to register and report new vulnerable devices. Since the DNS servers might also log queries the forensic scientist can check those as well for the same information.

4. ROADMAP

After the in depth description of the Mirai botnet's components the authors also proposed a roadmap serving as a guide to Mirai botnet forensics. The important note is that all of the components carry a lot of information about the whole process so the forensic scientist could start analysing from any of them. The key evidence for extracting important information are network packets, executables and live process analysis. While a lot of data can be gathered from any of the services the CNC server still contains the most information about the botnet, mostly because it's the only component which holds and actively tracks available botnet devices. Additionally the database server should be the second most important forensic artifact since it contains the static data about the botnet as well as the access history. Next in line are the Scan server and loader since the most information flows through them and they as well act upon infecting new devices. Lastly the DNS server and attacker's remote terminal are the least important since they mostly contain logs but don't have any other active role in the overall playbook.

5. CONCLUSIONS

The IoT world is vast and ever growing so, like in the early days of the internet, the new devices are not yet well protected. With the new communication techniques like 5G we try to connect more and more devices to the web. And with that we should strive to protecting them as much as we can, but since there is always a way to gain access to a device, we must also evolve the world of digital forensics. This article was a great start into the world of IoT since it proposed a

general roadmap how to progress through the infrastructure to collect information and take actions. But since it was limited to the Mirai setup it can only be used for this and similar branches of botnets. For future work we would need to identify more different types of IoT attacks and create more general roadmaps for handling these kinds of threats.

6. REFERENCES

[1] E. Bertino and N. Islam. Botnets and internet of things security. *Computer*, 50(2):76–79, 2017.

[2] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas. Ddos in the iot: Mirai and other botnets. *Computer*, 50(7):80–84, 2017.

[3] X. Zhang, O. Upton, N. L. Beebe, and K.-K. R. Choo. Iot botnet forensics: A comprehensive digital forensic case study on mirai botnet servers. *Forensic Science International: Digital Investigation*, 32:300926, 2020.

Parse and validate mobile forensic artifacts with Python

Merlin Baudou
Faculty of Computer and Information Science
University of Ljubljana
Vecna pot 113, 1000
Ljubljana, Slovenia
mb3319@student.uni-lj.si

Théotime Frognet
Faculty of Computer and Information Science
University of Ljubljana
Vecna pot 113, 1000
Ljubljana, Slovenia
tf08308@student.uni-lj.si

ABSTRACT
Mobile forensics is becoming more and more important as they are more and more present in our daily life. The data collected by our smartphones are a great help for investigators. However, it can be cryptic at first, which is why many researchers are working to understand what data is stored and how. To exploit this data it is necessary to build tools to visualize this data correctly and to make an efficient analysis. In this paper we will see how we can have access to the last image of an application, data allowing the construction of a timeline of the phone use. Finally, we will show two tools to exploit the databases present in the phones.

Keywords
Mobile Forensic, Artifact, Parsing,Databases

1. INTRODUCTION
The last few years have seen the development of tools to analyze the data present in our smartphones. These tools were not as advanced as those present on computers and the phones were not examined as thoroughly as computers. To be able to do a really efficient search, it was first necessary to understand in depth how the applications present on the phone stored the different data and what were the different ways to make a legal statement credible with the help of information retrieved from a phone.

This is how the researchers discovered that informations was not only stored in the Android databases but also in artifacts generated by the applications themselves, in particular with the help of artifacts in the form of XML files whose data is not accessible, which is why it is necessary to first parse the data before entering it into a database.

We'll first see some of these researchs by looking at how we can get information from Android data or Android application and then focus on a tool developed by Alexis Brignoni to parse data from Android artifact.

	account	doc_id	purchase_time
4	scubalover90210@gmail.com	com.cleanmaster.security	1524063410226
5	scubalover90210@gmail.com	com.google.android.talk	1524063478879
6	scubalover90210@gmail.com	com.instagram.android	1524063506996
7	scubalover90210@gmail.com	kik.android	1524063746230
8	scubalover90210@gmail.com	com.snapchat.android	1524063757141
9	scubalover90210@gmail.com	com.twitter.android	1524064586032
10	scubalover90210@gmail.com	com.whatsapp	1524064789396

Figure 1: Example of a com.vending.Android [3]

2. APPLICATION USAGE IN ANDROID
Android software stores a lot of data, so Investigators usually focus on some main points, web browsers, Chat App and Email because it's usually where useful evidence data can be found. But like computers there is a lot of data to get from applications, they create artifacts that can be used to understand what happened in a particular situation.

2.1 Application Analysis
The information to know when a particular application was installed is stored by the phone in the *com.vending.Android*. It is a folder that contains a database like the one in Figure 1, the database contains the account of the phone, with application installed and at what time.

But it is not enough to be sure that a user actually installed an application, because if another user uses his account on the device it will wipe the first database with the data from the other account [3].

The directory */data/system/usagestats* tells what application is in the foreground or what applications are in the background at a point of time. These data are stored in an XML file called epochtimestamp and are saved daily,weekly,monthly and yearly with an epoch timestamp. As well as some user interaction and configuration changes [3].

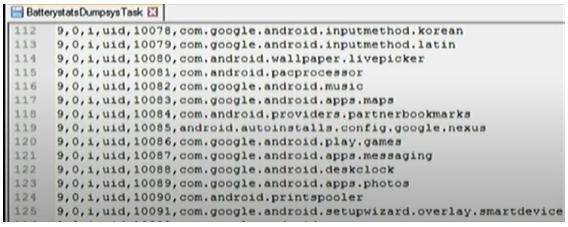


Figure 2: BatterystatsDumpsysTask.gz [3]

By researching into different appliclation data, we can find screenshots of what was on screen during the last access of any application before closing. The file *BatterystatsDumpsysTask.gz* contains among other things a list of applications with there UIDS. We can see an example of that in Figure 3

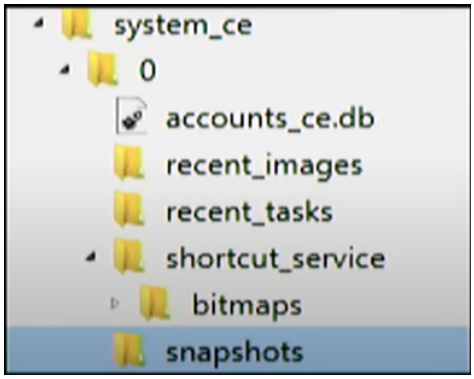


Figure 3: Example of a system_ce [3]

Then we can look in the directories of *system_ce*,the folder recent_images contains icon of application that were last used, we can see for example a Facebook logo and by looking into the database we can see the name of the icon activity with a number first, the task ID. The task ID is going to be the same for each application.

Let's go now in recent_tasks and we can see XML document with number that correspond of task id and inside this document we can find the UID that was in the battery stats dump manager. With that we tied application and verified that it is not a fake one.

In the snapshots directory, in system_ce we can find a lot of JPG with numbers, this numbers are task ID. These JPG files shows what was on the screen at the last time that application was used [3].

This shows that the forensic of a phone is not as easy as we think and not everything can be found in the classic databases. We can analyze what Android is storing inside his files about what a user is doing on his phone, and for example what is there in Digital Wellbeing?

2.2 Android Digital Wellbeing

The Digital Wellbeing is a dashboard for the usage data it collects. The device is tracking multiple data like the number of unlocks and the amounts of notifications received. Not only that but the time spent on the screen at any point of day, on each application.

Examining the data created by the application is quite easy as it is well documented and contained in SQlite database. But to get a timeline is not as easy as it looks. We'll be examining the table of events that is contained in /data/data/com.google.Android.apps.wellbeing/databases, it contains the timestamps (Unix Epoch) of what the application did, described by event code. By linking with the other databases we can quite easily get timestamps linked with the app and what is going on with each app.

	_id	timestamps	package_name	type	eventtype
1	1	2020-02-17 13:18:56	android	27	DEVICE_STARTUP
2	2	2020-02-17 13:19:02	com.android.settings	1	ACTIVITY_RESUMED
3	3	2020-02-17 13:19:02	com.android.settings	2	ACTIVITY_PAUSED
4	4	2020-02-17 13:19:02	com.google.android.setupwizard	1	ACTIVITY_RESUMED
5	5	2020-02-17 13:19:03	com.google.android.setupwizard	19	FOREGROUND_SERVICE START
6	6	2020-02-17 13:19:03	com.google.android.setupwizard	12	NOTIFICATION
7	7	2020-02-17 13:19:03	com.google.android.setupwizard	2	ACTIVITY_PAUSED
8	8	2020-02-17 13:19:03	com.google.android.pixel.setupwizard	1	ACTIVITY_RESUMED
9	9	2020-02-17 13:19:03	com.google.android.setupwizard	23	ACTIVITY_STOPPED
10	10	2020-02-17 13:19:03	com.android.settings	23	ACTIVITY_STOPPED
11	11	2020-02-17 13:19:06	android	12	NOTIFICATION
12	12	2020-02-17 13:19:10	com.google.android.apps.tycho	12	NOTIFICATION
13	13	2020-02-17 13:19:14	com.google.android.pixel.setupwizard	2	ACTIVITY_PAUSED
14	14	2020-02-17 13:19:14	com.google.android.setupwizard	1	ACTIVITY_RESUMED
15	15	2020-02-17 13:19:14	com.google.android.setupwizard	2	ACTIVITY_PAUSED
16	16	2020-02-17 13:19:14	com.google.android.setupwizard	1	ACTIVITY_RESUMED
17	17	2020-02-17 13:19:14	com.google.android.setupwizard	2	ACTIVITY_PAUSED

Figure 4: SQL query output of Wellbeing databases [2]

With this query (that can be export to .csv for example) we can see a lot of what is happening with this phone. Let's take a look at the figure 4. We can see that the device was switched on at 13:18:56, that the settings was the first app resumed, then the user switched to a setup wizard, and it goes on. With a longer query we could see for example how downloads are automatically done in the foreground without the user doing anything. By doing cross-examination with other application inside the phone investigators could corrobate what is going at which point of time on the phone.

However, we can note a few things like the fact that the database will not keep data about application that were deleted, or that the database would not have any entry of *com.exampleApp* after the exampleApp was deleted.

If investigators discovered that the Digital Wellbeing was turned off, it can be turned on and the last ten days of data are available. If the Digital Wellbeing is turned off it delete all the data that were in his databases. [2]

Now we will see a tool to extract easily all data that were presented and other.

3. XLEAPP TOOL

From the last part we saw that mobile phones contains a lot of informations about users, connections and a lot of data, usefull for digital forensics investigation.
This is why Alexis Brignoni decided to work on it.

3.1 History

He started parsing mobile informations using Python scripts. At first he wrote few scripts for parsing iOS mobiles data, but he started to gather a lot of different informations and needed to organize it. A friend suggested him to gather all these informations into a single framework that would be easier to use. iLEAPP was born : iOS Logs, Events, And Properties Parser. He had already wrote a few Android parsing scripts before creating, not so long after them, ALEAPP : Android Logs Events And Protobuf Parser. Today the xLEAPP frameworks are still updated regularly.

3.2 Goal

This tool was created with the purpose of helping digital investigations that are needing mobile forensics tools. This tool is an independant open source framework which makes it affordable for everyone. This tool is used to validate commercial DFIR (Digital Forensics & Incident Response) tools, and to help forensics examiners that can't afford a full suite of commercial DFIR tools for investigating Android and iOS systems.

This tool was intended to be used in several steps of an investigation. It can be used as triage tool for example, to maximize the time spent when searching for what to focus the investigation on. It can also be used in lab, in research, in validation steps of investigation, and finally for automation.

3.3 How it works

These frameworks are compatible with every PC OS (Tested on Windows, Mac OS and Linux, but works on everything that runs Python), and accept compressed files (.tar/.zip) or a directory which is the image of the mobile phone to investigate. From these, it will collect various artifacts, and parse them to find the attached useful data.

3.3.1 Artifacts collected

Android and iOS have different user artifacts, but the information gathered is quite the same.

- Mobile Installation Logs
- Application Usage and Activities
- Settings and builds data and logs
- Call and SMS history
- Notifications content
- Media and accounts
- Browser History
- Celular Wireless Information

This is a non exhaustive list of what these tools are capable to collect, and there is even more to come, considering that there are people still working on it.

These artifacts are collected from various databases (that depends on OS) and are parsed using Python scripts coded by Alexis Brignoni and various of his collaborators.

3.3.2 Data Parsing

To start parsing, there is two different options : Command line or graphic interface. Python needs a lot of different libraries here that are used to search and parse all the different artefacts.

```
Alexiss-MacBook-Pro:ALEAPP abrignoni$ python3 aleapp.py -h
usage: aleapp.py [-h] -t {fs,tar,zip} -o OUTPUT_PATH -i INPUT_PATH

ALEAPP: Android Logs, Events, and Protobuf Parser.

optional arguments:
  -h, --help            show this help message and exit
  -t {fs,tar,zip}       Input type (fs = extracted to file system folder)
  -o OUTPUT_PATH, --output_path OUTPUT_PATH
                        Output folder path
  -i INPUT_PATH, --input_path INPUT_PATH
                        Path to input file/folder
```

Figure 5: Help section of ALEAPP in cmd [1]

Using the command line, is really useful for automation (figure 5). The type of file examined need to be specified, as well as where it is situated and where the report should go.

But the easiest way of using any of the two frameworks is buy using the gui (figure 6). With the interface, the same informations than in command line can be chosen, but there is also a selection of the different artifacts that the framework can parse.

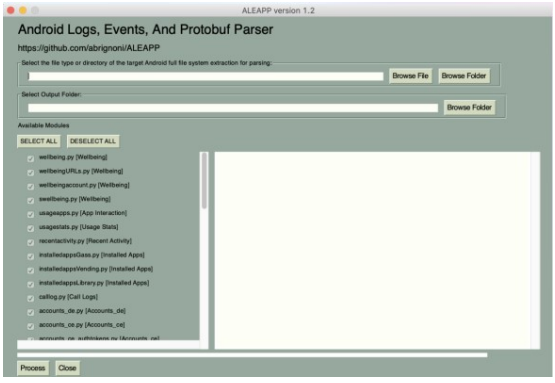


Figure 6: GUI of ALEAPP [1]

This tool needs to access different databases and files that are written in a variety of different languages. This is why Alexis Brignoni chose Python as his main language, so he could parse all these different files and databases written in SQL, Protobuf... Even Binary Files !

Here is an example of what the framework is doing behind the scenes (figure 7).

```
all_rows = cursor.fetchall()
usageentries = len(all_rows)
if usageentries > 0:
    report = ArtifactHtmlReport('Wellbeing events')
    report.start_artifact_report(report_folder, 'Events')
    report.add_script()
    data_headers = ('Timestamp', 'Package ID', 'Event Type')
    data_list = []
    for row in all_rows:
        data_list.append((row[1], row[2], row[4]))

    report.write_artifact_data_table(data_headers, data_list, file_found)
    report.end_artifact_report()

    tsvname = f'wellbeing - events'
    tsv(report_folder, data_headers, data_list, tsvname)
else:
    logfunc('No Wellbeing event data available')

db.close()
return
```

Figure 7: Example of xLEAPP's work [1]

After choosing an artifact to parse, collecting it and parsing it, the data then needs to be organized and reported. In the code screenshot (figure 7), we can see that we first get the result of the parse, then we populate the data's infos. In this example we collect the title of the artifact and the category to identify it on the report, then the headers to organize the data in the specified section, then all the said data connected. Next we put all the info we got into a TSV (Tabulation Separated Value) file, and return it to the final report. If the data collected have timestamps, it can also be collected and reported, helping with time activity for example. Same remark for KML information, which collects coordinate points (Latitude - Longitude)

4. CONCLUSIONS

In this paper, we have seen how Android stores different data, including UIDs and task IDs for each app that allow us to establish the veracity of the use of certain applications and furthemore that it is possible to retrieve the last thing an application was showing on the screen. Some basic functionalities like Android WellBeing allowed to build a first timeline of the use of the phone,even if it has to be confirmed with the study of other data to corrobate everything.

xLEAPP frameworks are powerful Digital Forensics tools that allow to obtain in a simple and readable way the data present in the various databases of an investigated phone, with the help of Python scripts centralized in them.

5. REFERENCES

[1] A. Brignoni. ileapp aleapp: Parse and validate mobile forensic artifacts with python. DFRWS 2020 USA, July 2020.

[2] J. Hickman. Walking the android (time)line. using android's digital wellbeing to timeline android activity. Available at https://thebinaryhick.blog/2020/02/22/walking-the-android-timeline-using-androids-digital-wellbeing-to-timeline-android-activity/ (22/02/2020).

[3] J. Hyde. Application and network usage in android. SANS DFIR, October 2018.

Forenzična analiza brskalnika Tor

Gregor Novak
Univerza v Ljubljani
Fakulteta za računalništvo in informatiko
gn0212@student.uni-lj.si

Jaša Kšela
Univerza v Ljubljani
Fakulteta za računalništvo in informatiko
jk6445@student.uni-lj.si

Žan Jaklič
Univerza v Ljubljani
Fakulteta za računalništvo in informatiko
zj8850@student.uni-lj.si

POVZETEK

V današnjem času varnost in anonimnost v spletu postajata vedno bolj relevantni. Uporabniki skušajo ostati anonimni z uporabo brskalnikov, kot je Tor. Njegov namen je zakriti uporabnikovo lokacijo in dejanja pred analizo prometa v internetnem omrežju. Uporaba Tora pa po drugi strani forenzikom oteži sledenje kriminalnih internetnih aktivnosti. V analiziranem članku so se avtorji osredotočili na forenzične artefakte, ki jih za seboj pusti uporaba Tora in ne toliko na spletne aktivnosti, katerim je primarno namenjen.

V članku najprej opišemo omrežje Tor in kako je le-to realizirano. Nato opišemo še brskalnik in podatke, ki jih Tor shranjuje. Zatem na kratko navedemo rezultate obstoječih raziskav. V nadaljevanju podamo metodologijo, uporabljeno v analiziranem članku, rezultate, ki so jih raziskovalci dosegli in glavne ugotovitve, do katerih so prišli. Povzamemo, da sta omrežje in brskalnik Tor namenjena predvsem omogočanju anonimnosti uporabnika v spletu, ne pa tudi zakrivanja uporabe brskalnika v osebnih napravah. Predlagamo, da bi bila izvedba podobne raziskave koristna tudi na sistemih Unix.

Ključne besede

Tor Browser Bundle, brskalnik Tor, forenzična analiza, analiza registrov, podatkovni artefakti

1. UVOD

Brskalnik Tor omogoča anonimno brskanje po svetovnem spletu. Uporaba brskalnika je v današnjem času relativno močno razširjena, brskalnik pa je velikokrat uporabljen tudi s strani kriminalcev za dostop do globokega spleta. Zaradi tega je brskalnik lahko velikokrat vključen v digitalni preiskavi. Dobro razumevanje delovanja brskalnika, kot tudi njegovega omrežja in puščanja sledi v datotečnih sistemih lahko tako pripomore k izčrpnejši in kvalitetnejši forenzični preiskavi.

Pregledali smo članek A forensic Audit of the Tor Browser Bundle [5], ki je bil leta 2019 predstavljen v reviji Digital Investigation. V poglavjih 2 in 3 bralca na kratko uvedemo v omrežje Tor in forenziko brskalnikov. V poglavju 4 predstavimo pregledano preteklo delo, ki se navezuje na povzet članek. Nato v poglavjih 5, 6, 7 in 8 predstavimo metodologijo in ključne ugotovitve avtorjev, ki so lahko v večji meri koristne pri digitalnih preiskavah. Na koncu v poglavju 7 povzamemo ugotovitve in predlagamo možnosti za nadaljnje raziskave.

2. OMREŽJE TOR

Tor definiramo, kot internetni omrežni protokol, cilj katerega je anonimnost omrežnega prometa TCP, ki temelji na porazdeljenem prekrivnem omrežju. Glavni namen uporabe protokola je anonimizacija brskanja po spletu. Omrežje je bilo narejeno leta 2002 v raziskovalnem laboratoriju ameriške mornarice. Trenutno projekt razvijata Agencija za napredne obrambne analize in Electronic Frontier Foundation.

Omrežje je sestavljeno iz prostovoljno vključenih strežnikov. To uporabnikom zagotavlja anonimnost v omrežju. Cilj omrežja Tor je prikriti originalni vir, od koder naj bi zahtevek prišel do strežnika. Promet na poti do strežnika preusmeri čez več različnih strežnikov in tako posledično prekrije sledi. Uporabnikov IP-naslov tako ni neposredno vezan na obiskano spletno mesto. [5]

Tor sestavljajo štiri glavne komponente: [4]:

- čebulasti posredovalniški strežnik,
- čebulasti usmerjevalnik,
- imeniški strežnik,
- skriti posredovalniki.

Čebulasti usmerjevalnik tvorijo trije deli: [4]:

- vhodni stražar,
- izhodni usmerjevalnik,
- vmesni usmerjevalnik.

Za izbiro ustreznega tipa komunikacije čebulasti usmerjevalnik uporablja deskriptor. Ta vsebuje tehnične podatke o samem usmerjevalniku in statistiko za izbiro optimalne poti. Vsak čebulasti usmerjevalnik v omrežju periodično pošilja svoj deskriptor imeniškim strežnikom. Posledično je stanje na omrežju venomer posodobljeno.

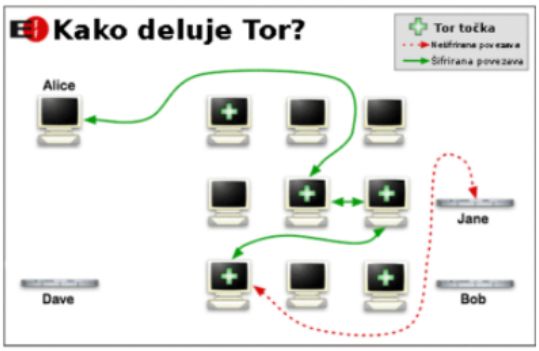
Deskriptor se pošlje v sledečih sitaucijah: [3]:

- Čebulasti usmerjevalnik je bil ponovno zagnan,
- od pošiljanja zadnjega deskriptorja je minil določen časovni interval,

- katero koli polje se je spremenilo, izjema je polje pasovna širina,
- v primeru spremembe pasovne širine, se mora ta spremeniti za vsaj faktor 2 glede na zadnji poslan deskriptor.

2.1 Izvedba protokola

Uporabnik omrežja ima nameščen čebulasti posredovalniški strežnik, ki s protokolom SOCKS posreduje naprej podatke protokola TCP v samo omrežje Tor. Z imenikom omrežja Tor pridobi spisek aktivnih čebulnih usmerjevalnikov. Sledi izbira tistih, ki bodo del prenosne verige. Na prvem členu se izvede enkripcija sporočila. Z naslednjim vmesnim členom se dogovori o ključu za kodiranje povezave. Postopek se ponavlja vse do zadnjega člena, kot vidimo na sliki 1. V zadnjem členu se izvede dekripcija paketa. Želeni strežnik posledično lahko razume paket. Šifriranje poteka zaporedno, za vsak paket ločeno [3].



Slika 1: Vizualni prikaz omrežja Tor [4].

2.2 Mostovi

Težava omrežja Tor bi lahko nastala v kolikor bi internetni ponudniki blokirali dostope do vseh znanih čebulastih usmerjevalnikov. Ta seznam je javno dostopen. Ustvarjalci omrežja Tor so dano težavo rešili s pomočjo mostov, pri katerih gre za čebulaste usmerjevalnike, ki jih ne najdemo v seznamih na imeniških strežnikih. Posledično jih ponudniki internetnih storitev ne morejo blokirati. Uporabnike mostov lahko opredelimo kot navadne uporabnike omrežja Tor, ki privzeto uporablja kriptirane povezave do imenskih strežnikov. Mostovi imajo lastne imenske strežnike imenovane avtoritete mostička. V kolikor uporabnik naleti na blokado, mora poiskati most, s katerega se nato lahko poveže v omrežje Tor [3].

3. FORENZIKA BRSKALNIKOV

Običajni brskalniki beležijo veliko količino podatkov med samo uporabo le-teh. Forenzika brskalnikov se ukvarja s pridobitvijo informacij iz brskalnika, ki nakazujejo na sumljivo spletno aktivnost osumljenca.

Informacije ponavadi pridobimo iz sledečih artefaktov brskalnikov [1]:

- zgodovina brskanja,
- zaznamki,
- predpomnilnik,
- prenosi,
- prijave,
- zbrane ikone (ang. favicons),
- samo izpolnitveni obrazci.

3.1 Brskalniki

Različni brskalniki hranijo podatke v različnih oblikah. Brskalnika Google Chrome in Mozilla Firefox podatke hranita v podatkovni bazi SQLite, medtem ko različica brskalnika Microsoft Edge pred implementacijo z ogrodjem Chromium hrani podatke v bazi ESE. Datoteke se načeloma lahko nahajajo na različnih lokacijah, odvisno od operacijskega sistema.

Za zagotovitev zasebnosti večina spletnih brskalnikov omogoča način brez beleženja zgodovine (ang. incognito mode). V navedenem načinu brskalnik ob zaprtju izbriše shranjene informacije glede brskanja. Izkaže se, da se tudi v tem primeru ob uporabi pravilnih metodologij do danih podatkov lahko dostopa. Ob testiranju večih brskalnikov se je kot najbolj varen način brez beleženja zgodovine pokazal način brskalnika Mozilla Firefox [6].

3.2 Časovni podatki

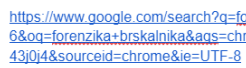
Digitalni forenzik mora biti pozoren na vse brskalnike, ki jih je osumljenec uporabljal. Največjo vrednost pri preiskavi brskalnikov najpogosteje predstavljajo časovni podatki. S pomočjo teh lahko forenzik sestavi časovno premico dejanj osumljenca. Pri tem mora biti pozoren na časovne formate posameznih brskalnikov in na časovni pas osumljenca, kajti vsi brskalniki hranijo časovne podatke v različnih fomatih v UTC-času.

Web Browser	Time Format
Internet Explorer	FILETIME: 100-ns (10^{-9}) Since January 1, 1601 00:00:00 (UTC)
Firefox	PRTime: microsecond(10^{-6}) Since January 1, 1970 00:00:00 (UTC)
Chrome	WEBKIT Time: microsecond(10^{-6}) Since January 1, 1601 00:00:00 (UTC)
Safari	CF Absolute Time: second Since January 1, 2001 00:00:00 (UTC)
Opera	UNIX Time: second Since January 1, 1970 00:00:00 (UTC)

Slika 2: Časovni formati v brskalnikih [6].

3.3 Analiza zgodovine iskanja

Zgodovina iskanja lahko pogosto predstavlja pomemben dokaz. Preiskovalec mora biti pozoren na iskane nize uporabnika ob uporabi spletnih iskalnikov. Te lahko pogosto razkrijejo dejanja osumljenca. Iskane nize lahko prepoznamo iz strukture naslova URL. Spodaj vidimo primer iskanja niza "digitalna forenzika" z iskalnikom Google.



Slika 3: Primer spletnega iskanja niza "digitalna forenzika" z iskalnikom Google.

Brskalniki uporabniku omogočajo izbris podatkov brskalnika. V kolikor je bila ta funkcija uporabljena je forenzična preiskava močno otežena.

Dokazi, ki jih za seboj pušča brskalnik so pomemben del digitalne forenzične preiskave. Pogosto so dokazi, ki nakazujejo na kriminalno dejanje razkriti prav iz preiskave brskalnika [6].

4. PRETEKLO DELO

Kljub temu, da brskalnik Tor omogoča visoko anonimnost uporabnikom v spletu, je bilo v predhodnih raziskavah ugotovljeno, da brskalnik v datotečnih sistemih računalnikov pušča sledi v obliki različnih podatkovnih artefaktov, ki so lahko pogosto kjučnega pomena v forenzičnih preiskavah. Za to, da so artefakti brskalnika dejansko obstoječi, so po večini krive različne omejitve operacijskih sistemov, ki se jim Tor ne more izogniti.

V nadaljevanju poglavja opišemo glavne ugotovitve preteklih del, na katere se opira tudi analiziran članek. Relevantne pretekle raziskave se osredotočajo predvsem na sledi, ki jih paket Tor Browser Bundle pušča v datotečnih sistemih različnih operacijskih sistemov.

4.1 Artefakti v operacijskem sistemu Windows

S pomočjo forenzične analize je bilo pokazano, da lahko Tor v datotečnem sistemu operacijskega sistema Windows pusti izsledljive dokaze [2]. Dokazano je bilo, da brskalnik pušča artefakte na naslednjih lokacijah znotraj sistema:

- mapa za "prefetch", C:\Windows\Prefetch\,
- predpomnilnik za predogledne sličice ("thumbnails"),
- ostranjevalna datoteka sistema Windows,
- registrska baza sistema, Windows registry.

4.2 Artefakti v operacijskih sistemih Debian GNU in Linux z namizjem GNOME

Tekom forenzične analize na sistemih Debian in Linux [7] je bilo pokazano, da lahko iz zgodovine uporabniške lupine */bin/bash* izluščimo, ali je uporabnik pognal brskalnik Tor.

Pokazano je bilo, da lahko v navideznem datotečnem sistemu GVFS, ki ga uporablja namizje GNOME, najdemo lokacijo datoteke tarball za Tor, ki nosi ime *tor-browser-gnu-linux-x86_64-2.3.25-5-dev-en-US.tar.gz*. Datoteka je bila najdena na lokaciji */home/runa/.local/share/gvfs-metadata/home*.

Datoteke tarball (končnica *.tar.gz*) se najpogosteje uporabljajo v sistemih UNIX in služijo, kot arhivi za stiskanje raznih datotek. V primeru zgoraj omenjene datoteke, gre najverjetneje za namestitveni paket programske opreme Tor Browser Bundle.

V datoteki na lokaciji */home/runa/.recently-used.xbel* se nahajajo podatki o nedavno uporabljenih datotekah. Avtorji so pokazali, da datoteka vsebuje ime navedene datoteke tarball, kot tudi datum in čas, ko je bil paket Tor Browser Bundle dodan, spremenjen in obiskan.

4.3 Artefakti v operacijskem sistemu OS X

Dokazano je bilo, da Tor pušča sledi v Applovemu sistemskemu dnevniku. Avtorji so navedli tudi, da tekom uporabe brskalnika Tor niso naleteli na nobeno težavo, vendar so v sistemskemu dnevniku vseeno našli Torove sledi, ki jih je pustila programska oprema The Crash Reporter sistema OS X, ki zbira informacije o aplikacijah, ki se sesujejo ali postanejo neodzivne [7].

Odkrito je bilo tudi, da je možno najti uporabniške nastavitve brskalnika, povezane s sistemskimi pisavami, preteklimi elementi ipd. v okviru datotek *.plist*, kamor aplikacije sistema OS X nasploh shranjujejo uporabniške preference.

4.4 Artefakti omrežnega prometa

V okviru forenzične preiskave, ki vsebuje brskalnik Tor, je pomembno, da poskusimo odkriti omrežni promet s strani brskalnika. Glede na to, da Tor ves promet kriptira, je potrebno uporabiti tehnike za detekcijo uporabe anonimnih posredniških strežnikov. V preteklih delih sta bili predlagani dve pravili za sistem Snort [2], za kateri avtorji trdijo, da sta sposobni odkrivanja omrežnega prometa s strani brskalnika Tor.

```
alert tcp $HOME_NET any -> $EXTERNAL_NET 80,443,9001,9030 (msg: "TOR client access
detected"; pcre:"/.*(Tor).+(client <identity>).*/i"; classtype:policy-violation; sid:50009;
```

Slika 4: Primer predlaganega pravila snort za odkrivanje omrežnega prometa brskalnika Tor [2].

5. METODOLOGIJA

Avtorji članka so se med forenzično preiskavo osredotočili predvsem na razširjanje že obstoječe literature o uporabi brskalnika Tor na operacijskem sistemu Windows 10 in dokazovanju, da uporaba protokola Tor na osebnem računalniku pusti sledi.

5.1 Pomisleki o metodologiji

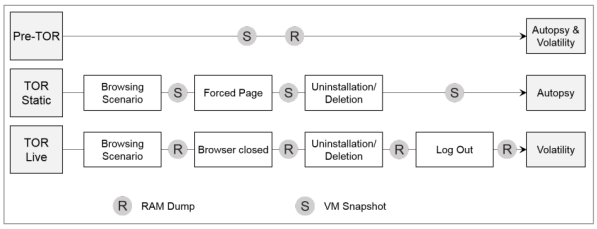
Glede na pretekle raziskave se je izkazalo, da je analiza programa med njegovim delovanjem učinkovita, saj lahko spremljamo interakcijo med programom in operacijskim sistemom. Tor je primarno osredotočen na onemogočanje pisanja na disk, analiza sistema med delovanjem programa pa bi potencialno lahko prinesla več informacij o seji brskalnika. V prejšnjih poskusih na starejših Torovih različicah je bilo mogoče najti več podatkov med delovanjem programa, kot

po njegovem zaprtju. Brskalnik Firefox v zasebnem načinu, ki ga uporablja tudi Tor, je sposoben ustaviti delujoče procese in iz pomnilnika izprazniti morebitne sledi o brskanju, ni pa znano, ali je tega sposoben tudi brskalnik Tor sam.

Z namenom razširjanja obstoječe literature o anonimnosti brskalnika Tor, so si avtorji zadali naslednja vprašanja:

- Ali Tor izbriše sledi o brskanju iz pomnilnika, ko končamo sejo?
- Ali lahko dokažemo uporabo Tora kadarkoli med/po njegovi uporabi ali njegovem izbrisu in odjavi uporabnika iz operacijskega sistema?
- Ali so podatki o brskalnem protokolu izsledljivi med delovanjem najnovejše različice brskalnika Tor?

Navedena vprašanja naj bi odgovorila, ali je uporabnikova anonimnost še vedno zaščitena v primeru, da je njegov računalnik zasežen kmalu po uporabi brskalnika Tor. Na sliki 5 so vidni koraki preiskave, ki pa je bila opravljena tudi na prenosni verziji Tora, ki naj bi glede na obstoječo literaturo puščala manj sledi.



Slika 5: Koraki forenzične preiskave [5].
Uporaba sistema pred namestitvijo brskalnika Tor.
Statična analiza in zajem diska virtualne naprave.
Analiza tekočega programa in zajem delovnega spomina.

Testni protokol za preizkušanje anonimnosti brskalnika Tor je zajemal brskanje po svetovnem spletu, vnašanje podatkov v obrazce ter prenašanje datotek in pretočnih vsebin. Naslovi spletnih strani in iskani vzorci so bili izbrani tako, da so vsebovali redke besedne zveze, kar je kasneje omogočalo iskanje besed brez lažno pozitivnih rezultatov.

5.2 Zajem delovnega spomina

Delovni spomin je bil izvožen iz navidezne naprave VirtualBox kot datoteka ELF 64, ki je bila potem analizirama z orodjem Volatility Framework. Uporabljeni so bili različni pripomočki za pridobivanje podatkov iz delovnega spomina:

- cmdline: argumenti ukazne vrstice za trenutno delujoče programe,
- dlllist: prikaz datotek DLL za posamezen proces,
- dumpfiles: izvleče predpomnjene datoteke,
- envars: izpiše okoljske spremenljivke procesov,
- pslist: s sledenjem jedrne strukture EPROCESS izpiše vse tekoče procese,
- pstree: izpiše vse procese v drevesno strukturo,
- shellbags: izpiše podatke shellbag iz registra,
- timeliner: naredi časovno premico iz artefaktov v pomnilniku.

Ena izmed pogostih forenzičnih praks je opazovanje tekočih procesov in njihov ID-vrednosti. Z uporabo ID-vrednosti lahko potem ta proces poiščemo s pomočjo orodja Volatility in sledimo procesu skozi čas. Avtorji so najprej analizirali brskalnik Tor med njegovim delovanjem, ugotovitve pa nato uporabili še v statični analizi, kjer so podrobneje preiskovali predvsem registre Windows.

6. REZULTATI

Avtorji članka so se forenzične analize pridobivanja podatkov o delovanju brskalnika Tor lotili na tri načine. Najprej so opravili analizo programa med njegovim delovanjem. Potem so se lotili statične analize, kjer so preiskovali stanje virtualnega računalnika po uporabi brskalnika Tor. Na koncu pa so podrobno pregledali še registre operacijskega sistema Windows in iskali morebitne sledi.

6.1 Forenzika med delovanjem programa

Delovni spomin virtualne naprave je bil zajet v štirih različnih trenutkih:

- po brskanju, ko je brskalnik še odprt,
- po zaprtju brskalnika,
- po brisanju brskalnika,
- po odjavi uporabnika.

Avtorji so ugotovili, da se po vsakem koraku številu sledi zmanjša, zato so se v rezultatih osredotočili predvsem na končno stanje.

Po izbrisu brskalnika Tor in odjavi uporabnika, so se procesi povezani z brskalnikom prenehali, še vedno pa je bil aktiven proces firefox.exe. Po iskanju ID-vrednosti procesa s pripomočkom pstree, so ugotovili, da ima ta proces izvor v namestitvenem direktoriju brskalnika Tor, ki uporablja Firefox. Prav tako pa so z uporabo pripomočka timeliner uspeli povezati začetni in končni čas procesa z brskalnikom Tor.

6.2 Statična forenzika

Posnetki virtualne naprave so bili zajeti v treh različnih trenutnkih, kot je vidno na sliki 5, vendar pa so se tudi v tem primeru avtorji osredotočili predvsem na analizo končnega stanja po izbrisu brskalnika Tor.

6.2.1 Iskanje besednih zvez

V prvem koraku so iskali besedne zveze, ki so jih uporabili pri brskanju v brskalniku Tor. Vse vnešene besede so bile najdene v datotekah NTUSER.DAT, ntuser.dat.LOG1 in MEMORY.DMP. Ena besedna zveza ni bila vsebovana v datoteki ntuser.dat.LOG1, ampak je bila najdena v nedodeljenem prostoru. Kasnejša analiza je pokazala, da so bili vsi podatki najprej vpisani v datoteko NTUSER.DAT,

nato pa kasneje kopirani v drugi dve datoteki. Datoteke z vsebovanimi besednimi zvezami so vsebovale celoten spletni naslov iskanih strani, ki jim je bila dodana besedna zveza "Tor Browser", kar očitno nakazuje na uporabo brskalnika Tor. Prav tako pa so vsebovale tudi pot do namestitvenega direktorija in ime uporabnika, ki je uporabljal brskalnik. Pri eni izmed iskanih besednih zvez pa je bilo mogoče razločiti tudi izhodni strežnik omrežja Tor. Pri izbrisu brskalnika Tor pa datoteke, ki so bile predhodno preko brskalnika prenešene s spleta, niso bile izbrisane.

6.2.2 *Nedodeljen prostor*

S pomočjo programa Autopsy je bilo iz nedodeljenega prostora mogoče izvleči nekaj artefaktov. Po iskanju besede v izklesanih datotekah je bilo v nedodeljenem prostoru mogoče najti več datotek s končnicami *dll*, *edb* in *reg*. Po podrobnejšem pregledu teh datotek je bilo mogoče izslediti spletni naslov obiskane strani, HTTP-glavo in prikazati, da je uporabnik uporabljal brskalnik v zasebnem načinu. Prav tako pa so v nedodeljenem prostoru ostale povezave do namestitvenega direktorija paketa Tor Browser Bundle.

6.3 NTUSER.DAT

Statična analiza je razkrila veliko količino artefaktov v datoteki NTUSER.DAT, ki shranjuje nastavitve uporabniškega profila. Avtorji so zato še bolj podrobno pregledali vsebino registrov z uporabo orodja Registry Explorer. Med preiskavo registrov so odkrili ključ shellactivities, ki vsebuje imena vseh obiskanih spletnih strani in pot do namestitvenega direktorija paketa Tor Broweser Bundle. Puščanje sledi so prav tako preizkusili z uporabo prenosljive različice brskalnika, naloženega na zunanji napravi. Tudi ta način je za seboj pustil veliko artefaktov. Iz enega registra je bilo možno razbrati tudi, da je bil brskalnik zagnan iz zunanje naprave. Poleg prej omenjenga ključa v registru, pa so našli artefakte tudi v zvočnem ključu, ki prikazuje, da je uporabnik v brskalniku poslušal zvočne vsebine.

Za kontrolo je bil uporabljen brskalnik Firefox v zasebnem načinu, ki je za seboj pustil podobne sledi kot brskalnik Tor.

6.4 Shellactivities

Podatki o brskanju so spravljeni v ne-volatilen spominski prostor. Uhajanje podatkov je neodvisno od različice brskalnika Firefox, kar nakazuje, da pri tem ključno vlogo igra operacijski sistem. Avtorji so ugotovili, da starejše različice sistema Windows v registrih hranijo manj podatkov. Največ artefaktov pa je bilo mogoče zaslediti v takratni najnovejši različici Windows, 1703. Bolj podrobno pa so pregledali tudi ključ sheallactivies, ki je sestavljen iz šestnajstiških vrednostih. Iz njega so uspeli razbrati razne časovne žige in obiskane spletne strani. Našli so tudi nekaj drugih vrednosti, ki bi lahko bile koristne, vendar pa jih niso raziskali v podrobnosti.

7. UGOTOVITVE

7.1 Forenzična metodologija

Predlagana forenzična metodologija za ugotavljanje uporabe brskalnika Tor temelji na ugotovitvah iz prejšnjih poglavij in sestoji iz naslednjih korakov:

- Izvozi delovni spomin in ga analiziraj s pripomočki psscan, pstree in timeliner, da ugotoviš uporabo brskalnika Tor in najdeš uporabnika, kar bo hkrati razkrilo tudi časovne žige,
- če je možno, izvozi datoteki pagefile.sys in hiberfile.sys,
- izvleci uporabnikov register NTUSER.DAT,
- preglej vsebino ključa shellactivities, v katerem išči besedi "Tor" in "Firefox",
- išči besedo "obfs4", v nedodeljenem prostoru, ki lahko razkrije IP-naslove iz omrežja Tor.

7.2 Forenzika med delovanjem programa

Preiskovanje sistema med delovanjem brskalnika Tor je razkrila procese, ki so povezani z njegovo uporabo. To so procesi firefox.exe, tor.exe in obfs4proxy.exe. S pomočjo orodja Volatility in imena procesov je potem možno najti namestitveni direktorij paketa Tor Browser Bundle, iz katerega je mogoče razbrati disk, kjer je bil brskalnik Tor uporabljen in uporabnika, ki ga je zagnal.

Po zaključku seje Tor uspe končati vse procese razen enega, kar oteži delo forenzikov pri dokazovanju uporabe brskalnika Tor, saj se izbrišejo številne okoljske spremenljivke, argumenti ukazne vrstice in datoteke DLL. Kljub temu pa lahko s pomočjo preostalega procesa firefox.exe izsledimo namestitveni direktorij paketa Tor Browser Bundle. Točen razlog, da se zadnji proces ne konča tudi po končani seji in odjavi uporabnika ni znan. Tukaj pa je vidna pomanjkljivost uporabe zunanje programske opreme, saj Tor nima vpliva na razvoj brskalnika Firefox in tako ne more predvideti posledic, ki ga ta ima na anonimnost uporabnikov.

7.3 Statična forenzika

Po statični forenzični analizi so avtorji ugotovili, da brskalnik Tor podatke zapisuje tudi na disk. Ti so vidni v obliki HTTP-glav, naslovov in imen obiskanih spletnih strani, ki jih je mogoče najti v registrih Windows. Večina podatkov o uporabi brskalnika Tor je tako mogoče zaslediti v datoteki NTUSER.DAT, ki forenzikom omogoča rekonstrukcijo aktivnosti uporabnika.

7.4 Artefakti brskalnika Firefox

V datoteki places.sqlite, ki hrani zgodovino brskalnika Firefox, je bilo mogoče zaslediti spletni strani Torovega bloga in navodil, ki se odpreta ob prvem zagonu brskalnika Tor po namestitvi. Iz teh dveh vnosov lahko sklepamo na uporabo brskalnika Tor.

8. KLJUČNE UGOTOVITVE

8.1 Forenzika med delovanjem programa

- Štiri procese lahko povežemo z uporabo brskalnika Tor, ko je ta odprt,
- lahko zaznamo uporabo proksija obfs4,
- le en proces ostane aktiven po zaprtju brskalnika,
- z uporabo programa Volatility lahko najdemo namestitveni direktorij in časovne žige,
- iz namestitvenega direktorija lahko najdemo uporabniško ime in lokacijo brskalnika na disku.

8.2 Statična forenzika

Preprečevanje pisanja na disk ni doseženo:

- Obnovimo lahko konfiguracijske in prenesene datoteke ter datoteke, povezane z brskalnikom,
- izsledimo lahko sledi v obliki HTTP-glav, imen in naslovov spletnih strani v registrih, sistemskih datotekah in nedodeljenem prostoru,
- izsledimo lahko izhodni strežnik omrežja Tor.

Varen in popoln izbris brskalnika Tor ni dosežen:

- Izsledimo lahko namestitveni direktorij brskalnika, datoteke SQLite, zaznamke, datoteke DLL,
- izsledimo lahko uporabo zasebnega načina brskanja, ki je povezana z brskalnikom Tor,
- najdemo lahko datoteko, ki hrani časovne žige zagonov programa.

9. ZAKLJUČEK

Glede na ključne ugotovitve sklepamo, da bi forenziki s podrobno analizo naprav bili zmožni dokazati uporabo brskalnika Tor, poskus njegovega izbrisa, kdaj je bil nazadnje zagnan, katere strani so bile z njim obiskane in kje se nahaja izhodni strežnik uporabljenega omrežja Tor.

Vredno je omeniti, da je bil v preiskavi raziskan le paket Tor Browser Bundle in pripadajoč brskalnik Tor, ne pa tudi omrežje Tor, katerega analiza bi lahko pripeljala še do dodatnih ugotovitev. Kljub temu, da so bili forenzični pristopi dokaj preprosti, npr. iskanje besednih zvez, je bilo mogoče pridobiti veliko količino podatkov o uporabniku in njegovem brskanju.

Avtorji so si anonimnost, ki jo Tor nudi, razlagali precej bolj široko, saj so pričakovali neke nestandardne akcije brskalnikov, kot je npr. odstranitev vseh prenesenih datotek po izbrisu brskalnika Tor. Glede na prebrano, je Tor zasnovan predvsem z namenom, da skrije uporabnika pred analizo prometa v internetnem omrežju, ne pa tudi, da uporabnika ščiti pred forenzično preiskavo lastnega računalnika. Nekatera pričakovanja avtorjev članka se zato zdijo prevelika. V članku je bilo dokazano, da nekaj artefaktov ni posledica uporabe Tor protokola, ampak je za njih odgovoren brskalnik Firefox, ki ga Tor uporablja. V primeru, da bi razvijalci Tora želeli zagotoviti anonimnost tudi na fizični napravi, bi bilo smiselno uporabiti lastno implementacijo brskalnika, saj uporaba zunanje programske opreme lahko pripelje do nepredvidljivih dogodkov, vendar pa, kot smo že omenili, to verjetno ni prvotna skrb razvijalcev Tor protokola V nadaljnih raziskavah se nam zdi smiselno preiskati še artefakte najnovejšega brskalnika Tor na sistemih Unix.

10. LITERATURA

[1] N. Bencherchali. Web Browsers Forensics. Dosegljivo: `https://nasbench.medium.com/web-browsers-forensics-7e99940c579a`. Dostopano: 5. 5. 2021.

[2] D. Dayalamurthy. Forensic Memory Dump Analysis And Recovery Of The Artefacts Of Using Tor Bundle Browser – The Need. *11th Australian Digital Forensics Conference. Held on the 2nd-4th December*, 2013 at Edith Cowan University:Western Australia, 2013.

[3] L. Demšar. Temni splet in kriminal. Diplomsko delo, Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko, 2016.

[4] V. Glavač. Identifikacija uporabnikov omrežja Tor. Magistrsko delo, Univerza v Ljubljani, Fakulteta za matematiko in fiziko, 2017.

[5] M. Muir, P. Leimich, and W. J. Buchanan. A Forensic Audit of the Tor Browser Bundle. *Digital Investigation*, 29:118–128, 2019.

[6] J. Oh, S. Lee, and S. Lee. Advanced evidence collection and analysis of web browser activity. *Digital Investigation*, 8:S62–S70, 2011. The Proceedings of the Eleventh Annual DFRWS Conference.

[7] R. A. Sandvik. Forensic Analysis of the Tor Browser Bundle on OS X, Linux, and Windows. *Tor Tech Report 2013-06-001*, 2013.