



Master thesis

Master's Programme in Network Forensics, 60 credits

Digital forensics of cryptocurrency wallets

Examiner: Mark Dougherty

Thesis in Digital Forensics, 15 credits

Halmstad 2022-05-20

Tomas Kovalcik

Abstract

The rise of cryptocurrencies over the past ten years was significant. Cryptocurrencies have changed the course of the monetary system and created a new way for people to communicate and exchange values. It started with the invention of the first successful cryptocurrency, Bitcoin, in 2008. After this moment, the explosion from which thousands of new cryptocurrencies emerged and from which a new field for digital forensics investigators happened. This thesis deals with the digital forensics of cryptocurrency wallets. The work aims to target applications running on Linux operating systems, and to introduce concepts and existing work on cryptocurrency wallets. Furthermore, the goal was to develop a tool for finding different artifacts that wallets leave on the systems. This work is divided into two parts, first encompasses an introduction and the related literature. The second part discusses the chosen methodology for this work and the result we achieved. The work is concluded with the final words in the Conclusions section.

Keyword: cryptocurrencies, wallets, artifacts, python

Acknowledgements

I would like to thank my supervisor for allowing me to do this freely and on my own, and my family and friends, without whose support I would not have been able to complete this work.

Table of contents

1	Introduction.....	7
2	Research Objective	8
2.1	The goal of the master's thesis.....	8
2.2	Limitations.....	8
3	Thesis Outline	9
4	Literature Review	10
4.1	Cryptocurrency wallets.....	10
4.1.1	Software wallet.....	10
4.1.2	Hardware wallet	11
4.1.3	Wallet related artifacts	11
4.2	Digital Forensics.....	12
4.2.1	Linux OS as a platform for cryptocurrency wallets	12
4.2.2	Regular Expressions.....	13
4.2.3	Forensic Algorithms.....	13
4.2.4	Forensics Investigation cycle	14
4.3	Related work.....	15
5	Methodology	16
5.1	Literature sources	16
5.2	Practical part.....	16
5.3	Development approach.....	17
5.4	Proposed design.....	17
5.5	Proposed testing	18
5.6	Summary	18
6	Results.....	19
6.1	Practical part – The application.....	19
6.2	Exodus vs Electrum wallet	21
7	Discussion.....	24
7.1	More on Exodus vs Electrum wallet	24
7.2	Our solution compared to the related work	24
8	Conclusion	25
8.1	Future work	26
8.2	Concluding words.....	26

References	I
Appendices.....	IV
Source code	IV
main.py	IV
regular_expressions.py	XII

1 Introduction

The rise of cryptocurrencies over the past ten years was significant, and right now, we are living in an age where cryptocurrencies have made their point. They entered our lives inadvertently, and now they are with us, and we are with them. The first successful cryptocurrency Bitcoin [1], introduced in 2008 and launched in January 2009, completely changed the course of the monetary system. It has created a new way for people to communicate and exchange values as well as it has separated money from the state and has given power back to the people. This breakthrough has defined the way people will do business in the upcoming years. The Bitcoin invention caused the explosion from which thousands of new cryptocurrencies emerged. Although monetary systems and financial institutions have been analyzed for many years now, cryptocurrencies create a new field for digital forensic investigators, as the distributed architecture of the new money makes this a challenge.

Definition of cryptocurrency is that it is a peer-to-peer digital currency that is exchanged by using specific principles of cryptography and can be used as any other fiat currency with respect to the regulation. As one of the principles, blockchain is a public ledger where all transactions are publicly recorded, effectively dismantling the function of centralized authorities [2]. The software which creates transactions is called a wallet. A cryptocurrency wallet holds keys to the funds locked on the blockchain. This software runs on personal computers, mobile phones, or dedicated hardware. From the digital forensics' investigation point of view, it is a fascinating field to explore from which lots of new challenges come up. Investigators always face new challenges when they get a device to identify, scan and discover new valuable facts and information from it that could help them move forward and create a better and more precise picture of what has happened and what the device is about.

There are also other reasons why investigation of cryptocurrencies is important. Despite the many advantages they provide, they are also often used for illicit doing such as money laundering, buying illegal drugs, weaponry, and armory, all happening in the underground economies. They are introducing new types of economic crime organizations that are hard to investigate using traditional investigation methods by the police and the financial, tax, or other authorities [3]. Furthermore, they can be used for funding terrorism of any kind, whether political or religious.

Digital forensics experts worldwide have discovered new challenges with this modern technology; therefore, innovative approaches, methodologies, and tools must be explored and implemented.

2 Research Objective

Many questions arise from the topic described above as well as from our current knowledge of cryptocurrency wallets. The wallets are an important part when operating with bitcoin or any other cryptocurrency. There are millions of wallets installed worldwide and left for investigation and information gathering. This leads us to ask the following questions below which are interesting enough for us to explore what can be found on the system once the wallet is installed and get educated about cryptocurrencies. This research will provide us with theoretical and practical knowledge and aid us in our future professional life.

Question 1: What are the weaknesses of cryptocurrency wallets?

This question urges us to look at the implementation of the wallets and what artifacts they leave on the system. In addition to that, this question aims to compare the different wallets.

Question 2: What are possible approaches for finding wallet artifacts?

This question aims to explore different methods for analyzing various artifacts left in the system. To answer this question, we must analyze existing software tools and develop our own tool to fully understand what the possibilities are when looking for artifacts.

2.1 The goal of the master's thesis

The goal of the master thesis is to provide an analysis of crypto-space, precisely wallets, which are used to store assets of specific cryptocurrencies. Another outlook of this thesis is to analyze existing forensics tools for exploring systems and detecting the presence of wallet files. The last step is to provide a tool for detecting the artifacts.

2.2 Limitations

Because of the limited time frame, this research was performed only on desktop wallet applications. The developed CLI application also targets only Linux Desktop cryptocurrency applications.

3 Thesis Outline

In this paper, we discuss approaches to digital forensics of cryptocurrency wallets, describe and analyze the concepts related to cryptocurrency wallets and propose our own methodology and solution to this problem. Later we discuss the results we have achieved, discussion, and the potential future work. The appendix contains the source code of the application.

This paper is organized as follows: Literature Review, Methodology, Results, Discussion, Conclusions, References, and Appendices.

4 Literature Review

This section will encompass information about cryptocurrency wallets and work that has already been done toward digital forensics investigation of cryptocurrency wallets.

4.1 Cryptocurrency wallets

When one wants to enter the world of cryptocurrencies, a wallet is needed. A wallet is a crucial component for end-users to operate and make changes on the blockchain. The primary purpose of the wallet is to send and receive funds, create transactions, and broadcast them to the network. Unlike conventional wallets that we use in everyday life, cryptocurrency wallets do not directly hold the funds. A token which is a product of cryptocurrency, is stored on the blockchain and that can be accessed and altered only by the correct keys which are stored in the cryptocurrency wallet. So, in that sense, we can think of a cryptocurrency wallet as a keychain [2].

Primarily, two keys are generated at the wallet creation. A private key and a public key. They are an intrinsic component of cryptocurrencies based on blockchain networks. When talking about public and private keys, we are indirectly referring to Public Key Cryptography (PKC) [4]. The public key is an address to which funds can be sent, and the private key is used to send or spend the funds. The key management can be categorized as hosted, managed by a third party, and non-hosted, managed by the user wallets. The former is considered insecure as the funds are owned by the provider of the wallet, i.e., crypto exchanges [5].

Another important feature of the wallet is its anonymity. In the case of a non-hosted wallet, no personal information is required, and there are no traces of who is the owner of the wallet. In the case of hosted wallets, lots of information is usually required [5]. This type of wallet is no concern of this thesis.

4.1.1 Software wallet

A software wallet is usually a mobile or desktop application. These types of wallets are connected to the Internet to make transactions. Although the developers of these wallets implement various safe protection precautions to mitigate attack vectors as much as possible, they do not provide a one hundred percent secure way of storing the keys in an offline manner. Therefore, when a computer or a mobile device is attacked by a virus or a hacker, there is a certain risk that the keys will be exploited [5]. In Figure 1, two wallets are presented in their desktop versions. They offer different user interfaces, but they both aim for the same goals.

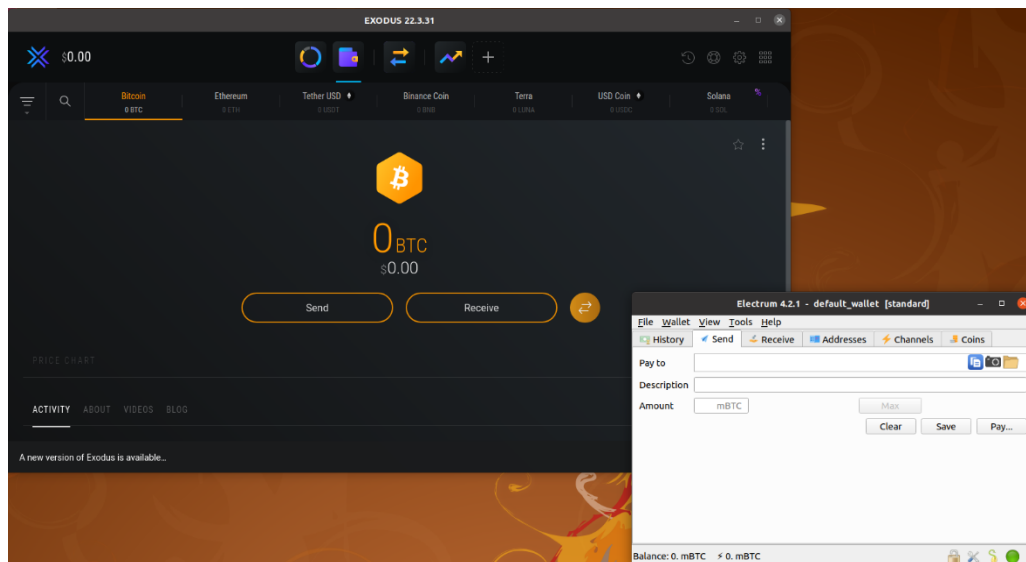


Figure 1 Exodus (left) and Electrum (right) software wallets installed on the Ubuntu virtual machine. Exodus wallet features more complex user interface and provides more functionality. Electrum wallet has simpler user interface and work only with bitcoin whereas Exodus is a multicurrency wallet.

4.1.2 Hardware wallet

A hardware wallet refers to a physical device such as a USB or another specialized device. It still needs to be connected to the Internet to make transactions, but the keys are kept securely and are not exposed to a live internet connection. It is much harder to hack a hardware wallet because its keys are kept in an offline environment [5].

4.1.3 Wallet related artifacts

When we are dealing with software wallets, they often must create configuration files, logs, and other temporary files to function. These files encompass lots of information that can be valuable to a digital investigator. Here we list the most interesting and common artifacts found on a seized computer.

- A private key unlocks and proves that the user is the owner of the funds. Bitcoin's private key is a 256-bit number and can be represented in many different forms [2].
- A public key is derived from the private key and does not need to be saved in the wallet. It is a public part of the key pair which cannot spend funds, but it is still a necessary cryptography component.
- Bitcoin address is derived from the public key and is represented by 58 characters, using Base58Check encoding, Base58 number system [2].
- Wallet's related files, like logs and temporary files, are usually present and are interesting artifacts, and they can reveal more about the wallet's state [6].

- Mnemonic phrases are used in hierarchical deterministic (HD) wallets and are used to recover all the keys that have ever been created. Finding traces of such artifacts would lead to a huge discovery. The format of a mnemonic phrase is 12 or 24 English words, each with a minimum length of 3 characters up to 8 characters [6].

Below in the Table 1 are the most common types of keys and their prefixes.

Type	Hex Version prefix	Base58 prefix
Bitcoin Address	0x00	1
P2SH Address	0x05	3
Bitcoin Testnet	0x6F	m or n
Private key WIF	0x80	5, K or L
Encrypted Private Key	0x0142	6P
Extended Public Key	0x0488B21E	xpub

Table 1 Base58Check version prefix and encoded result examples [2] of the keys. The most common types are listed. The "type" column holds different types of addresses, followed by Hexadecimal version prefix and Base58 prefix. These are important to know for pattern matching.

There are other address types such as bitcoin vanity address, private key WIF compressed, encrypted private key, mini private key, and a new type of bitcoin address, bitcoin bech32 [7]. Bech32 is a segwit address format also known as "bc1 addresses". Bech32 is more efficient with the block space and it has been in use since October 2020. The Bech32 address format is supported by many popular wallets, and is the preferred address scheme today. Bitcoin vanity address is a normal bitcoin address, except it can be customized by the user. The purpose of the customization is to make it more personal and identifiable [8].

4.2 Digital Forensics

There are specific ways to perform digital forensics on digital evidence, and it all depends on what is to be examined. In the following subsections, we will discuss platforms, techniques, and approaches with more general applications.

4.2.1 Linux OS as a platform for cryptocurrency wallets

Linux operating system is the most used operating system in the world. Its versatility, stability, and adroitness to perform specific tasks have been appreciated by many, and its uses have been found useful in the crypto space. There are Linux distributions specifically developed to target the crypto space [9]. Linux has certain security advantages over other operating systems, and therefore it is often used for hosting different services, including cryptocurrency wallets. Unix-like systems are less prone to malware attacks than Windows machines from a security standpoint. One of the reasons is the highly configurable system that it offers to the user. Programs cannot be run

without appropriate permissions, and their openness gives more people a chance to look at the source code and eliminate any potential threats. The system package updates are coming at a more frequent pace, and in the case of a found exploit, the fix comes within a few hours.

4.2.2 Regular Expressions

A regular expression, shortened regex, is a sequence of symbols and characters expressing a pattern to be searched for within a specific data. This technique is handy and efficient for all kinds of problems, from programming language compilers to text processing tools and editors. In forensic analysis, regexes are helpful as they help investigators quickly find desired patterns and give them a chance to focus on relevant evidence [10]. As the theory behind regular expressions is vast, we will explain it by a specific example.

Let's consider a regular expression [13][a-km-zA-HJ-NP-Z1-9]{25,34} matches strings that correspond to the following. We will start from the right side of the expression. The '{25,34}' matches the previous token between 25 and 34 times, as many times as possible, giving back as needed. The actual format of the string is specified inside the square brackets. The first set of square brackets matches a single character in the list '13'. The second set of brackets represents another part of the pattern. All hyphenated characters represent ranges. The range 'a-k' matches a single character in the range between a and k. The range 'm-z' matches a single character in the range between m and z. The same applies to digits, so any number in the range '1-9' will be valid. The regular expression matches all bitcoin addresses in the legacy format.

For example, the string 1BvBMSEYstWetqTFn5Au4m4GFg7xJaNVN2 would be matched by the regex. Of course, this example is straightforward, and some patterns are much harder to understand and write. This explanation is sufficient for understanding the regular expressions that will be used in our case.

4.2.3 Forensic Algorithms

Investigation experts must use proper techniques to seize, and document found artifacts. In digital forensics, there are many ways to achieve this. One of the most common approaches is to use hashing algorithms to get the fingerprints of seized files. The standard forensics algorithms are MD5, SHA256, SSDEEP [11].

MD5 message-digest algorithm produces a 128-bit hash value represented in a text format as a 32-digit hexadecimal number. The advantage of this algorithm is that it is fast and memory efficient. The disadvantage is its incompetence in producing collision-free hashes [11].

SHA Secure Hash Algorithm family consists of several hash functions capable of producing hash values that are between 224 bits and 512 bits. SHA-256 is the algorithm that is used in most blockchain applications. It remains a collision-free algorithm and should be a preferred option [11].

SSDEEP is the opposite of a cryptographic hash function because the resulting hash changes only a little bit when the input is changed by one bit. Its purpose is to check how two files are similar. This is useful for checking if the source code of two applications is similar [11].

The presented forensics algorithms are a critical component of the chain of custody. The chain of custody refers to the evidence that can be used in court. One part of the chain of custody is an evidence log. An evidence log contains all details from the investigation together with the information on who is in control of the evidence. Typically, an evidence log will contain information such as the date and time, the collected evidence, the name of the investigator, and the location. In the case of computer forensics, information about the specific machine must be included, for example, hostname, CPU chip, memory information, and IP address [12], and other Indicators of Compromise (IOCS) that could be interesting for an investigator [13].

4.2.4 Forensics Investigation cycle

In every investigation, certain steps are followed. It is an important part of the investigation, and at the same time, it simplifies the work and process of the investigation team. Figure 2 below presents the five main steps of digital investigation which are used in everyday life of a forensic investigator.

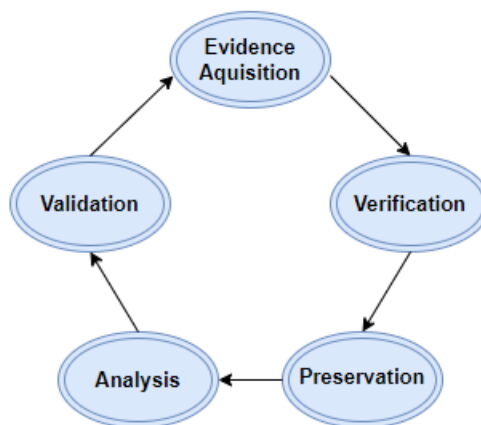


Figure 2 Forensics investigation cycle [14]. There are five steps in the investigation starting with Evidence Acquisition going to Verification, Preservation, Analysis and finally Validation. These steps are common in all kinds of forensics.

Evidence acquisition is the first step which includes a forensic image of the evidence without any tampering of the original data. Verification is the second step in which cryptographic hashes are created for the evidence and then compared with the original data. A hash acts as a signature and assures

the integrity of the evidence [14]. On Linux machines, a utility called GnuPG [15] can be used to forge a signature. Following is the preservation of the evidence, thus collected data must be protected from any factors that could amend the evidence. Analysis is usually the fourth step in forensic investigation and includes artifact extraction. This step also answers five questions who, when, what, where, and why. As the last step, validation is performed to ensure sound and valid results [14].

4.3 Related work

Various works deal with the forensics of cryptocurrency applications. Here we mention some works that deal with mobile and desktop applications and describe their purpose and findings.

In the [16], they propose a methodology for preserving crypto wallets at the crime scene. In the paper, they presented a detailed process flow to preserve digital evidence. They described all the steps and performed simulated tests.

The [17] analyzes the security risks of Android-based cryptocurrency wallets. They describe the threat model and security goals and analyze the attack surface and vectors. The test was done on two digital currency wallet applications, and they found some serious security risks. This work helps us realize what the weaknesses of those applications are.

The work [18] analyzes the security of commonly used Android cryptocurrency applications. The vulnerabilities outlined by OWASP are examined and tested.

In [19] authors present some pitfalls of crypto wallets and how they can be exploited. They discuss different methods on how to exploit the specific type of wallet. This paper contributes to the crypto wallet industry by showing potential bugs in the wallets.

In the work [6], the authors present an open-source tool for live forensic and postmortem analysis. They discuss several types of wallets and the artifacts they leave on the system once installed. Using the tool, they can produce a list of target artifacts obtained from a forensic analysis of different Bitcoin clients and web wallets installed on Windows 7 and 10. The paper aims for the same goals as our work; however, they target only Windows operating system.

5 Methodology

Our focus is to develop a tool for forensic analysis of cryptocurrency wallet artifacts on Linux operating systems. At the time of writing this thesis, no publications were found that would discuss digital forensics of cryptocurrency wallets on the Linux OS. In our understanding, it is necessary to do more research and develop more tools for this platform. Due to a limited time, we will focus only on desktop cryptocurrency wallets, excluding browser wallets. Additionally, we focused primarily on bitcoin wallets, although the examined wallets are multi-currency wallets.

Two main scientific methods were chosen, a literature study and an experiment in a form of programmed application. In the next subsections we discuss the chosen approaches.

5.1 Literature sources

The literature described in the Related work section was mostly obtained through scientific and conference papers available in common databases such as Institute of Electrical and Electronics Engineers (IEEE) and Association for Computing Machinery (ACM). Apart from conference papers, we gathered information from books and white papers available for free on the Internet. Many software engineering sites such as StackOverFlow and StackExchange were used for the practical parts. The literature research was done in the initial weeks of the summer semester, from the second half of January till the end of February. The focus was on gathering information about cryptocurrency wallets. Only after doing comprehensive research were we able to ask the research questions.

5.2 Practical part

In the practical part of this thesis, we focused on finding artifacts that wallets could leave on the system. The implementation of the tool is in the Python programming language. The choice of the language has been affected by multiple reasons, the most important one is knowledge of the language and its simplicity. We chose simplicity before speed and efficiency. The code runs on the Linux OS. The tests were performed on virtual machines running the latest Ubuntu LTS version. The program's output is a summary of found files and their paths. The application is controlled via a command-line interface, and it provides a help menu for the user. The digital forensics was done in a post-mortem fashion, and thereby it was assumed that the computer which is suspected of the cryptocurrency activity is started and unlocked, and our program could be loaded onto it. The program scans the computer's filesystem looking for the artifacts described in section 4.1.3.

5.3 Development approach

As a development process, we chose an iterative process where we did small iterations, and each iteration has been evaluated and discussed. This approach allowed us to work on smaller pieces of functionality. Furthermore, this way eliminated potential bugs and irrecoverable code structures. On top of this, time savings and stress have been put to a minimum.

Another objective of ours was to support the chain of custody. The integrity of forensic evidence must be ensured when collecting the evidence, and throughout the entire handling and analysis [11]. To achieve the integrity of the seized files, there must be a method of creating a fingerprint of the file.

Forensics Algorithms are an essential method for forensic investigation. We decided to use the SHA256 algorithm to support the chain of custody for this tool. The advantages of this algorithm were described in the Literature section.

Table 2 presents regular expressions that will be used in the application. These expressions represent different formats of what the artifact could look like. Some of the expressions have been acquired from [6] and tested at [20].

BITCOIN_P2SH	b"[a-km-zA-HJ-NP-Z1-9]{24,33}"
BITCOIN_BECH32	b"bc1[a-zA-HJ-NP-Z0-9]{25,59}"
BITCOIN_LEGACY	b"[13][a-km-zA-HJ-NP-Z1-9]{25,34}"
EXTENDED_PUBLIC_KEY	b"xpub[a-km-zA-HJ-NP-Z1-9]{107,108}"
WIF_PRIVATE_KEY	b"5[a-km-zA-HJ-NP-Z1-9]{50}"
WIF_COMPRESSED_PRIVATE_KEY	b"[KL][a-km-zA-HJ-NP-Z1-9]{51}"
ENCRYPTED_PRIVATE_KEY	b"6P[a-km-zA-HJ-NP-Z1-9]{56}"
EXTENDED_PRIVATE_KEY	b"xprv[a-km-zA-HJ-NP-Z1-9]{107,108}"
MINI_PRIVATE_KEY	b"S[a-km-zA-HJ-NP-Z1-9]{29}"
ELECTRUM	b"[eE]lectrum"
EXODUS	b"[eE]xodus"

Table 2 Regular expressions used in the application. The list can be easily expanded to match other types of patterns. The regular expressions are encoded in a "bytes" format (that is expressed by the b before the actual expression.)

5.4 Proposed design

As outlined in the previous subsection 5.2, the application was controlled by the command line. It was designed to provide a help menu and multiple options from which the forensics investigator will be able to choose an appropriate setting to match the target environment. The code was written in an object-oriented way following the newest programming techniques and paradigms according to Python standards. The code contains functions only from the standard library, so that no third-party libraries is required. The only prerequisite would be to have Python3.8+ installed on the target machine.

5.5 Proposed testing

The implementation was tested on Ubuntu OS. Two distinct types of wallets were tested. Namely Exodus and Electrum. The first one is proprietary, and the latter one is open source. We wanted to see if there is any difference between open-source and proprietary in terms of security and obscurity of crypto artifacts. Another merit was how many artifacts and information about the presence of a bitcoin client we can acquire and seize. The performance of the implemented tool and the results are to be evaluated by a human and summarized in the Results section.

5.6 Summary

In this section, we have proposed our methodology for this thesis. We have described the sources of our literature and practical part and proposed an approach to successfully reach this thesis's goals. Furthermore, the design and testing proposals have been introduced and consequentially developed. In the next section, we will go over the results achieved in the development and mention the limitations we dealt with.

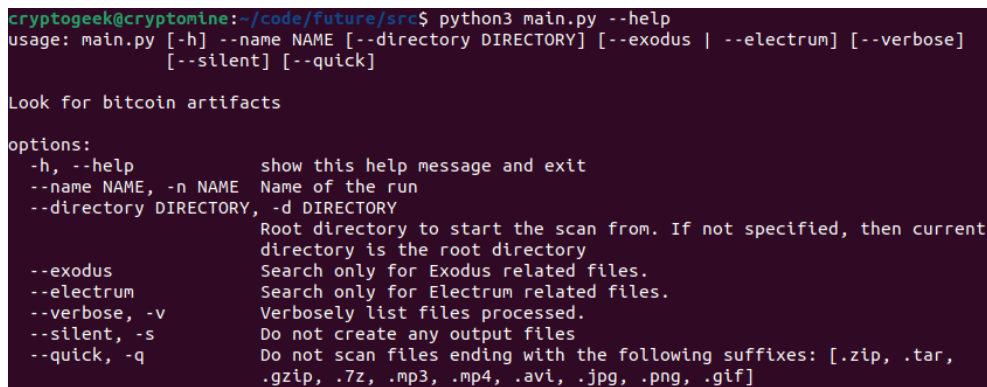
6 Results

This section discusses the achieved results and explains the limitations and differences between the chosen wallets.

6.1 Practical part – The application

As we described in the methodology section, our environment consisted of a virtual machine running Ubuntu 20.04 with the preinstalled python3.8. The virtualization of the OS gave us a clean environment each time we performed a new test or when the machine was polluted with unnecessary files generated by the tests. To further save our troubles, we created a snapshot of a virtual machine with all the necessary components already installed, so we would not have to repeat the same steps every time.

The implemented CLI application offers a user menu to provide better user experience. What the help menu looks like can be seen in the Figure 3 below.



```
cryptogeek@cryptomine:~/code/future/src$ python3 main.py --help
usage: main.py [-h] --name NAME [--directory DIRECTORY] [--exodus | --electrum] [--verbose]
               [--silent] [--quick]

Look for bitcoin artifacts

options:
  -h, --help            show this help message and exit
  --name NAME, -n NAME  Name of the run
  --directory DIRECTORY, -d DIRECTORY
                        Root directory to start the scan from. If not specified, then current
                        directory is the root directory
  --exodus              Search only for Exodus related files.
  --electrum            Search only for Electrum related files.
  --verbose, -v         Verbosely list files processed.
  --silent, -s          Do not create any output files
  --quick, -q           Do not scan files ending with the following suffixes: [.zip, .tar,
                        .gzip, .7z, .mp3, .mp4, .avi, .jpg, .png, .gif]
```

Figure 3 The help menu of the application shows different options that can be used while running the application. The options `--name` and `--directory` are mandatory and must be specified before running. Other options are optional and options `--exodus` and `--electrum` are mutually exclusive.

The system's scan can be initiated from any directory that the user specifies as one of the parameters. The process recursively traverses the filesystem from the root directory (the user specifies the root directory) and reads all files in binary mode unless specified otherwise with the quick option. It reads the files line by line and checks all patterns that are specified as regular expressions. Apart from the scanning process, the snapshot of currently running processes is taken and examined, searching for processes related to Exodus or Electrum wallets. In addition to that, command history is also examined. Any suspicious commands related to the wallet operation are reported if there are commands present that could be associated with the cryptocurrency wallets. Each run of the application creates a set of reports and compressed files that hold potential public or private keys that could be used to access the assets. Other file reports hold information about the files and individual pattern matches. As an example, see Figure 4 which shows the program's output. The program prints to the standard output of all files that it

has explored. In the end, the summary of the run is displayed, with some interesting information that could be useful for the forensic investigator.

```
cryptogeek@cryptomine:~/code/future/src$ python3 main.py --name test1 -q -s --electrum --dir ~/.electrum
Exploring file: /home/cryptogeek/.electrum/config [2 matches]
Exploring file: /home/cryptogeek/.electrum/daemon_rpc_socket [0 matches]
Exploring file: /home/cryptogeek/.electrum/blockchain_headers [0 matches]
Exploring file: /home/cryptogeek/.electrum/recent_servers [0 matches]
Exploring file: /home/cryptogeek/.electrum/daemon [0 matches]
Exploring file: /home/cryptogeek/.electrum/certs/xtrum.com [18 matches]
Exploring file: /home/cryptogeek/.electrum/certs/E-X.not.fyi [25 matches]
Exploring file: /home/cryptogeek/.electrum/certs/electrum.jochen-hoenicke.de [0 matches]
Exploring file: /home/cryptogeek/.electrum/certs/electrum.hsmiths.com [12 matches]
Exploring file: /home/cryptogeek/.electrum/certs/elx.bitske.com [6 matches]
Exploring file: /home/cryptogeek/.electrum/certs/e2.keff.org [0 matches]
Exploring file: /home/cryptogeek/.electrum/certs/btc.ocf.sh [10 matches]
Exploring file: /home/cryptogeek/.electrum/certs/bitcoins.sk [15 matches]
Exploring file: /home/cryptogeek/.electrum/certs/node.degga.net [0 matches]
Exploring file: /home/cryptogeek/.electrum/certs/hodlers.beer [18 matches]
Exploring file: /home/cryptogeek/.electrum/certs/horsey.cryptocowboys.net [0 matches]
Exploring file: /home/cryptogeek/.electrum/certs/bitcoin.lukechilds.co [9 matches]
Exploring file: /home/cryptogeek/.electrum/certs/128.0.190.26 [9 matches]
Exploring file: /home/cryptogeek/.electrum/certs/eal.coincited.net [10 matches]
Exploring file: /home/cryptogeek/.electrum/wallets/default_wallet [119 matches]

*** RUN SUMMARY ***

System information: sysname=Linux, nodename=cryptomine, release=5.15.0-27-generic, version=#28-Ubuntu S
MP Thu Apr 14 04:55:28 UTC 2022, machine=x86_64

Files containing bitcoin related patterns: 12
Total number of found patterns: 253
Total number of unique patterns: 177
Wallet/bitcoin processes running: True
Wallet/bitcoin command used: False
electrum wallet exists: True

TIP: To save this output to a file, use output redirection (e.g. python3 main.py --name test -q > quick
_scan.txt)

*** SUMMARY END ***
```

Figure 4 Test run. All the explored files are listed along with number of patterns matches in each file. Lastly, the run summary is presented with the most interesting data about the run such as host system information, total number of found patterns and files.

The example of the reports can be seen in Figure 6 and Figure 7. The file holds information about the artifacts that were found during the scanning. The file is constructed from three columns, the column file, which holds the path to the file on the scanned system, the type of the pattern, and the hit column, which represents the found artifact. A hit represents a string that matches one of the regular expressions described in the Methodology section.

Running command: `python3 main.py --name test1 --electrum --dir ~/.electrum/` gives the following results and outputs. The standard output can be seen in the figure above. Three files were created after this command ran. See Figure 5

```
-rw-rw-r-- 1 cryptogeek cryptogeek 55151 apr 30 18:50 test1-20220430-185032-artefacts.csv
-rw-rw-r-- 1 cryptogeek cryptogeek 3617 apr 30 18:50 test1-20220430-185032-hashed_files.csv
-rw-rw-r-- 1 cryptogeek cryptogeek 7103 apr 30 18:50 test1-20220430-185032-process_snapshot.txt
-rw-rw-r-- 1 cryptogeek cryptogeek 48511 apr 30 18:50 test1-20220430-185032-.zip
```

Figure 5 Files created after running the main.py script. The names of the files are composed of the name specified by the user, in this case it is "test1" and a timestamp that is generated by the code. This helps to identify the time the code was run. Four files are created once the code is finished, two csv files, one text and zip file containing all the files containing artifacts.

The figures below represent a small sample of what the reports look like. The information is saved in CSV format mainly because of better reading and better support in Python.

/home/cryptogeek/.electrum/wallets/default_wallet	nduaqctcawex73qrj4t8hmu98nuqfna	BITCOIN_VANITY_ADDRESS
/home/cryptogeek/.electrum/wallets/default_wallet	bc1q3qyw2lnduaqctcawex73qrj4t8hmu98nuqfna	BITCOIN_BECH32
/home/cryptogeek/.electrum/wallets/default_wallet	zprvAZ9oh28EvjY3UR1rVmEz9K1Say8Pb	BITCOIN_VANITY_ADDRESS
/home/cryptogeek/.electrum/wallets/default_wallet	3UR1rVmEz9K1Say8PbHDURJRggNSWf52Svn	BITCOIN_LEGACY
/home/cryptogeek/.electrum/wallets/default_wallet	52Svnwa61VGEK5sK4h6fQbRvWd1VnFXL0MSxKuCAgn95vQEJoXs	WIF_PRIVATE_KEY
/home/cryptogeek/.electrum/wallets/default_wallet	K1Say8PbHDURJRggNSWf52Svnwa61VGEK5sK4h6fQbRvWd1VnFXL	WIF_COMPRESSED_PRIVATE_KEY
/home/cryptogeek/.electrum/wallets/default_wallet	Say8PbHDURJRggNSWf52Svnwa61VGE	MINI_PRIVATE_KEY
/home/cryptogeek/.electrum/wallets/default_wallet	zpub6n9A6Xf8m76Lgu6KbnmzWSx8BzxsZjwKnXMHdk8QQZRobGidYoWn7QMAJCx9wL	BITCOIN_VANITY_ADDRESS
/home/cryptogeek/.electrum/wallets/default_wallet	Lgu6KbnmzWSx8BzxsZjwKnXMHdk8QQZRobGidYo	WIF_COMPRESSED_PRIVATE_KEY
/home/cryptogeek/.electrum/wallets/default_wallet	Sx8BzxsZjwKnXMHdk8QQZRobGidYo	MINI_PRIVATE_KEY
/home/cryptogeek/.electrum/wallets/default_wallet	zprvAcHAAcVa6ckNtZtVTPtoLbzHXA2V	BITCOIN_VANITY_ADDRESS
/home/cryptogeek/.electrum/wallets/default_wallet	16feq8d5qTQn1EzLerkdjxg4o4AKivsUR4	BITCOIN_LEGACY
/home/cryptogeek/.electrum/wallets/default_wallet	5qTQn1EzLerkdjxg4o4AKivsUR4zPhztVjWf7xFAeAp6w45vRN	WIF_PRIVATE_KEY

Figure 6 Artifacts found and gathered in CSV format (3 columns - file, hit, type). The first column file shows the full path to the file on the system, hit represents the found pattern and the last column type tells what type of address/pattern it is.

/home/cryptogeek/.electrum/certs/elix.bitske.com	4a30958bae8ab8b104400263d8f3951952b6b834baaf5336e42680d5f7fd3bbf
/home/cryptogeek/.electrum/certs/electrumx.electronicnewyear.net	8e132d869e1c3339611c7e63aeeabe4882321bb89a759ddd46585d2c89ab8d7e
/home/cryptogeek/.electrum/certs/73.92.198.54	58148fa77e51264cfc3b6e3dc0aecd1d0cb229f813eb8316ee6854a8603d2bf
/home/cryptogeek/.electrum/certs/128.0.190.26	e2852e5dec75dbaf661fd38e9c9aa2cf399ff5d0a25d7d55bc87283aa5d42ee
/home/cryptogeek/.electrum/certs/vmd84592.contaboserver.net	0ca4e6800e5e57cbd3c1f90c47149853bcf5b6314b494bd2fbc9ba1aed2a2ce7
/home/cryptogeek/.electrum/wallets/default_wallet	1adca1df2f1e6bf412bd6fa80453d63f957ababf22b5aa63b18591ed46c02c1c
/home/cryptogeek/.electrum/certs/gd42.org	62569abb739399b4e4e3c55d65141e5ac1a26b9777e08849e9e5de081c700589
/home/cryptogeek/.electrum/certs/vmd71287.contaboserver.net	ca74d218194b9c78489c93cdf6cbf2da42eda55effec6c454412afb4f8d2cde5
/home/cryptogeek/.electrum/certs/electrum.necrypto.io	1ffb8d1eced59dea048cedf4cb6593ea4e93f3c92e2f5d6fc4678f46aeb0841f

Figure 7 Files containing the found artifacts with their respective hash value (following the chain of custody). The first column "file" shows full path to the file containing artifacts and second column "hash" holds SHA256 hash of the specific file.

6.2 Exodus vs Electrum wallet

First, Electrum and Exodus wallets were installed separately, each in its virtual box, and then their installation directories were examined. Electrum wallet gives an option to the user to choose not to encrypt the wallet. It also allows users to create multiple wallets on the same system under the same user account. The default installation directory is in the ~/.electrum directory. Exodus wallet is stricter about how many wallets one can have under one account. Exodus allows only one wallet per user account. The default installation directory is in the ~/.config/Electrum directory.

Next step was to run our application. The results gave a little bit of an impression of which wallet is more discreet. Both wallets were installed with the default setting without setting a password. Both wallets created files as time passed, and we identified them mostly as temporary files and log files. As shown in figures (Figure 8, Figure 9, Figure 10, Figure 11), the wallets were installed on the same machine, and the tests were run twice for each wallet. The tests were executed after each other with a few minutes difference.

In the Electrum wallet data directory, the first test reported 13 files as sources of bitcoin artifacts out of 20 explored. The second test reported 14 files as the source of bitcoin artifacts out of 21. The total number of patterns found was 141 and 149 for each test, respectively. In the exodus wallet data directory, five files were identified as sources of bitcoin artifacts in the first test run and 8 in the second. The total number of patterns found in them was 10 and 13 for

each test, respectively. The most common patterns found in the files were bitcoin bech32 and bitcoin legacy addresses.

```
*** RUN SUMMARY ***

Test executed at: 15:29:22
System information: sysname=Linux, nodename=cryptomine, release=5.15.0-27-generic, version=#28-Ubuntu SMP Thu Apr 14 04:55:28 UTC 2022, machine=x86_64

Total number of explored files: 20
Files containing bitcoin related patterns: 13
Total number of found patterns: 141
Total number of unique patterns: 101
Wallet/bitcoin processes running: True
Wallet/bitcoin command used: False
electrum wallet exists: True

Three most common types of addresses:
BTCOIN_BECH32: 60 hits
BTCOIN_LEGACY: 41 hits
BTCOIN_P2SH: 28 hits

Results saved to: ['test4-20220511-152918-process_snapshot.txt', 'test4-20220511-152918-hashed_files.csv', 'test4-20220511-152918-artefacts.csv', 'test4-20220511-152918-.zip']

TIP: To save this output to a file, use output redirection (e.g. python3 main.py --name test -q > quick_scan.txt)

*** SUMMARY END ***
```

Figure 8 Electrum scan summary (1st run) shows the results. Pay attention to the numbers. Total explored files 20 and 13 of them contained bitcoin related patterns. Interesting to see is wallet/bitcoin processes running and the three most common types of addresses found during the scan.

```
*** RUN SUMMARY ***

Test executed at: 15:34:31
System information: sysname=Linux, nodename=cryptomine, release=5.15.0-27-generic, version=#28-Ubuntu SMP Thu Apr 14 04:55:28 UTC 2022, machine=x86_64

Total number of explored files: 21
Files containing bitcoin related patterns: 14
Total number of found patterns: 149
Total number of unique patterns: 109
Wallet/bitcoin processes running: True
Wallet/bitcoin command used: False
electrum wallet exists: True

Three most common types of addresses:
BTCOIN_BECH32: 60 hits
BTCOIN_LEGACY: 43 hits
BTCOIN_P2SH: 32 hits

Results saved to: ['test4-20220511-153427-process_snapshot.txt', 'test4-20220511-153427-hashed_files.csv', 'test4-20220511-153427-artefacts.csv', 'test4-20220511-153427-.zip']

TIP: To save this output to a file, use output redirection (e.g. python3 main.py --name test -q > quick_scan.txt)

*** SUMMARY END ***
```

Figure 9 Electrum scan summary (2nd run) Pay attention to the numbers. Total explored files 21 and 14 of them contained bitcoin related patterns. Interesting to see is wallet/bitcoin processes running and the three most common types of addresses found during the scan.

```

*** RUN SUMMARY ***

Test executed at: 15:47:51
System information: sysname=Linux, nodename=cryptomine, release=5.15.0-27-generic, version=#28-Ubuntu SMP Thu Apr 14 04:55:28 UTC 2022, machine=x86_64

Total number of explored files: 143
Files containing bitcoin related patterns: 5
Total number of found patterns: 10
Total number of unique patterns: 4
Wallet/bitcoin processes running: True
Wallet/bitcoin command used: False
exodus wallet exists: True

Three most common types of addresses:
BITCOIN_LEGACY: 7 hits
MINI_PRIVATE_KEY: 3 hits

Results saved to: ['test2-20220511-154750-process_snapshot.txt', 'test2-20220511-154750-hashed_files.csv', 'test2-20220511-154750-artefacts.csv', 'test2-20220511-154750-.zip']

TIP: To save this output to a file, use output redirection (e.g. python3 main.py --name test -q > quick_scan.txt)

*** SUMMARY END ***

```

Figure 10 Exodus scan summary (1st run). Pay attention to the numbers. Total explored files 143 and only 5 of them contained bitcoin related patterns. This shows the discretion of the wallet. Interesting to see is wallet/bitcoin processes running and the three most common types of addresses found during the scan.

```

*** RUN SUMMARY ***

Test executed at: 15:52:36
System information: sysname=Linux, nodename=cryptomine, release=5.15.0-27-generic, version=#28-Ubuntu SMP Thu Apr 14 04:55:28 UTC 2022, machine=x86_64

Total number of explored files: 161
Files containing bitcoin related patterns: 8
Total number of found patterns: 13
Total number of unique patterns: 7
Wallet/bitcoin processes running: True
Wallet/bitcoin command used: False
exodus wallet exists: True

Three most common types of addresses:
BITCOIN_LEGACY: 10 hits
MINI_PRIVATE_KEY: 3 hits

Results saved to: ['test2-20220511-155236-process_snapshot.txt', 'test2-20220511-155236-hashed_files.csv', 'test2-20220511-155236-artefacts.csv', 'test2-20220511-155236-.zip']

TIP: To save this output to a file, use output redirection (e.g. python3 main.py --name test -q > quick_scan.txt)

*** SUMMARY END ***

```

Figure 11 Exodus scan summary (2nd run). Pay attention to the numbers. Total explored files 161 and only 8 of them contained bitcoin related patterns. The discretion is still maintained. Interesting to see is wallet/bitcoin processes running and the three most common types of addresses found during the scan.

Exodus wallet, in its default settings, did not reveal its configuration. Instead, everything was in binary format so that a human could not read. On the other hand, Electrum wallet kept its configuration in JSON format, unencrypted, thus revealing much more information, including the 12-word seed. Using a password, the file was encrypted, thus impossible to read.

7 Discussion

Our application was developed under the name future. This name reflects our two main points of view. The view on cryptocurrency technology. In the name, we refer to our belief that it is the future of payments and transactions. In addition, the name future also refers to the way the code is structured and to the language that was used in the development. We used the latest version of Python, but the code is compatible with Python3.8+. The code is written in an object-oriented manner using essential OOP primitives such as classes and encapsulation. The code is decoupled, individual parts are affecting each other minimally, and the addition of new code structures is possible. The application provides a simple command-line interface where the user can choose from multiple options.

7.1 More on Exodus vs Electrum wallet

As we mentioned in section 6.2, wallets tend to create new files over their existence on the system. This fact makes it inappropriate to make a comparison based solely on found files containing traces of related artifacts. Thus, a more appropriate solution is to compare a contamination value which is a ratio between the reported files and all the explored files. In the case of the Electrum wallet, approximately fifty percent of explored files contained cryptocurrency-related artifacts. In contrast, in the case of Exodus wallet, the percentage value was a lot smaller, six percent in the first test and twelve percent in the second. These numbers are not definitive, but they give an approximate overview of which wallet is more exposed.

7.2 Our solution compared to the related work

Our solution targets Linux machines and has unique features that no other solution available on the Internet has, such as process examination and command history examination. This work presents the literature and digital forensic perspective to analyzing crypto-related artifacts on Linux machines. Compared to the work done in [19], our solution targets desktop applications instead of web browser wallets and is, in essence, wallet vendor agnostic. It looks for any traces of cryptographic hashes that could represent potential keys to crypto funds. Additionally, it checks running processes and examines whether a bitcoin wallet is running in the background. On the other hand, in [19], they have implemented a RAM analysis, the functionality discussed in the future work section.

8 Conclusion

This work dealt with digital forensics of cryptocurrency wallets. The main contribution of this thesis is the command-line application for discovering bitcoin-related artifacts such as private and public keys and addresses. Initially, this document introduces cryptocurrencies and their advantages and disadvantages from the author's point of view. Then the document takes a more technical view of cryptocurrencies and discusses the technology from an engineer's point of view, backed by relatable sources, explaining the various concepts and technologies behind the invention. Different types of cryptocurrency wallets are explained and illustrated. The most important concepts about forensics investigation are discussed, and the practical part sticks to them. Finally, from the literature section, the related work is briefly described. The methods we have chosen for this project worked and brought us a clear and definitive goal. The practical part of the work was developed in Python under Linux operating system. The thesis is divided into two parts, whereas the practical part makes up about 65 percent of the content from our point of view.

The thesis objective was also to answer two research questions that arose from the initial analysis. Although the questions have been subtly answered over the span of the document, we summarize them here in the thesis conclusion. The first question dealt with the weaknesses of cryptocurrency wallets. The two wallets selected by us conformed to a high standard of safety. We chose this question to see whether there is a difference in how the data are protected in a proprietary wallet versus an open-sourced wallet. From the results, we must conclude that the proprietary wallet protected its data better and did not expose the data in a human-readable format, even when unprotected by any password. However, this comparison does not necessarily mean that proprietary software will always perform better.

The second research question dealt with finding an approach to finding wallet artifacts. The related work in the literature review helped us realize what different methods exist. In our solution, we have decided to use regular expressions for searching patterns in files. In addition to that, we realized that running processes could be helpful too in the case of forensics investigation, so we are checking them and looking for possible running processes associated with cryptocurrency wallets. In our case, we are searching for Exodus and Electrum processes. It is an effective way of finding bitcoin traces; however, as every coin has two sides, this approach also has two sides, and we will address the bad side in the subsection related to future work. There are other possible ways to examine a computer. However, these were out of the scope of this document. Instead, we will discuss them in the section focusing on future work.

8.1 Future work

As mentioned in the previous subsection, there is room for improving the application by expanding its functionality or modifying its current behavior. One of the additions is to implement memory fingerprinting for later analysis. The idea is to make a copy of the contents of RAM and analyze them later. Memory often encompasses interesting information, such as a piece of crypto-related information, including wallet passwords. Implementing something like that requires us to have the knowledge in low-level computing and hardware memory architecture. There are frameworks like [21] which could be integrated into the tool or at least as a step in the digital investigation process. Another improvement of the application could be to refactor and modify current behavior of pattern matching.

Further improvements that could be made in the future are implementing a query function, which would query the existing databases of public addresses recorded on the blockchain. This is more of an automation improvement as this could be performed manually on sites such as [22] [23].

8.2 Concluding words

This master thesis allowed us to look at cryptocurrencies and their related and important components from both theoretical and practical perspectives. Studying the theory and our results gives us an advantage in the future, where we suspect an immense emergence of cryptocurrencies into everyday life. From an engineering perspective, we gained much knowledge of different technologies, which will aid us in our future work.

References

- [1] S. Nakamoto, "bitcoin.org," 2008. [Online]. Available: <https://bitcoin.org/bitcoin.pdf>. [Accessed 1 3 2022].
- [2] A. M. Antonopoulos, Mastering Bitcoin: Unlocking Digital Cryptocurrencies, 1st ed., O'Reilly Media, Inc., 2014.
- [3] D. Kavallieros, D. Myttas, E. Kermitsis, E. Lissaris, G. Giataganas and E. Darra, Dark Web Investigation, B. Akhgar, M. Gercke, S. Vrochidis and H. Gibson, Eds., Springer International Publishing, 2021.
- [4] "ledger.com," The Ledger Company, [Online]. Available: <https://www.ledger.com/academy/blockchain/what-are-public-keys-and-private-keys>. [Accessed 03 03 2022].
- [5] S. Saurabh, S. Mahesh and B. Sunil, "Cryptocurrency Wallet: A Review," in *2020 4th International Conference on Computer, Communication and Signal Processing (ICCCSP)*, Chennai, India, 2020.
- [6] S. Zollner, K.-K. R. Choo and N.-A. Le-Khac, "An Automated Live Forensic and Postmortem Analysis Tool for Bitcoin on Windows Systems," *IEEE Access*, vol. 7, pp. 158250-158263, 2019.
- [7] "en.bitcoin.it," [Online]. Available: <https://en.bitcoin.it/wiki/Bech32>. [Accessed 12 04 2022].
- [8] "academy.bit2me.com," [Online]. Available: <https://academy.bit2me.com/en/what-is-a-vanity-address/>. [Accessed 30 04 2022].
- [9] T. Hyde, "blackdown.org," [Online]. Available: <https://www.blackdown.org/best-linux-distros-for-mining-cryptocurrency/>. [Accessed 10 03 2022].
- [10] Computer Hope, "computerhope.com," [Online]. Available: <https://www.computerhope.com/unix/regex-quickref.htm>. [Accessed 10 05 2022].
- [11] M. Spreitzenbarth, Mastering Python Forensics, Packt Publishing, 2015.
- [12] "justcriminallaw," [Online]. Available: <https://www.justcriminallaw.com/criminal-charges->

- questions/2020/08/26/chain-custody-important-criminal-case/.
[Accessed 28 04 2022].
- [13] "crowdstrike," 13 05 2021. [Online]. Available:
<https://www.crowdstrike.com/cybersecurity-101/indicators-of-compromise/>. [Accessed 28 04 2022].
- [14] Y. Gorasiya, "medium.com," [Online]. Available:
<https://medium.com/cyversity/digital-forensics-investigation-steps-f06a82c2c4ac>. [Accessed 15 04 2022].
- [15] "gnupg.org," [Online]. Available: <https://gnupg.org>. [Accessed 18 03 2022].
- [16] S. K. Taylor, A. Ariffin, K. A. Z. Ariffin and S. N. H. S. Abdullah, "Cryptocurrencies Investigation: A Methodology for the Preservation of Cryptowallets," in *2021 3rd International Cyber Resilience Conference (CRC)*, Langkawi Island, Malaysia, 2021.
- [17] D. He, S. Li, C. Li, S. Zhu, S. Chan, W. Min and N. Guizani, "Security Analysis of Cryptocurrency Wallets in Android-Based Applications," vol. 34, no. 6, pp. 114 - 119, 2020.
- [18] A. . R. Sai, J. Buckley and A. L. Gear, "Privacy and Security Analysis of Cryptocurrency Mobile Applications," 2019.
- [19] T. Bui, S. P. Rao, M. Antikainen and T. Aura, "Pitfalls of open architecture: How friends can exploit your cryptocurrency wallet," in *Proceedings of the 12th European Workshop on Systems Security*, Dresden, Germany, 2019.
- [20] "regex101," [Online]. Available: <https://regex101.com/>.
- [21] "volatilityfoundation/volatility3," [Online]. Available:
<https://github.com/volatilityfoundation/volatility3>. [Accessed 02 05 2022].
- [22] "blockchain.com," [Online]. Available:
<https://www.blockchain.com/explorer>.
- [23] "blockchair.com," [Online]. Available: <https://blockchair.com/>.
- [24] "en.bitcoin.it," [Online]. Available:
https://en.bitcoin.it/wiki/BIP_0039. [Accessed 26 02 2022].

Appendices

Source code

main.py

@dataclass

class Match:

file: str

hit: str

type: str

@dataclass

class HashedFile:

file: str

fingerprint: str

class Processor:

def __init__(self):

self.host_information = os.uname()

def get_host_information(self) -> str:

return (

f"sysname={self.host_information.sysname}, "

f"nodename={self.host_information.nodename}, "

f"release={self.host_information.release}, "

f"version={self.host_information.version}, "

f"machine={self.host_information.machine}"

)

@staticmethod

def examine_process_snapshot(process_snapshot:

subprocess.CompletedProcess) -> bool:

"""

*check current running processes and return number
of indications that some crypto process might be running in
the background*

"""

if re.search(EXODUS, process_snapshot.stdout) or re.search(
ELECTRUM, process_snapshot.stdout

):

return True

return False

```

@staticmethod
def get_running_processes() -> subprocess.CompletedProcess:
    # for later analysis
    return subprocess.run(["ps", "-eo", "pid,args"], capture_output=True)

```

```

@staticmethod
def examine_command_history() -> bool:
    path = Path("~/bash_history")
    if path.expanduser().exists():
        try:
            with open(path.expanduser(), "rb") as file:
                for line in file:
                    if re.search(EXODUS, line) or re.search(ELECTRUM, line):
                        return True
        except PermissionError as err:
            # change color to red when printing error
            fail = "\033[91m"
            endc = "\033[0m"
            print(f"{fail}Failed reading {path}. Reason {err}{endc}")
    return False

```

```

class FileOperator:

```

```

    """

```

A job of this class is to provide an interface to perform operations on files that were created during digital forensic compression, writing to files (CSV, txt).

```

    """

```

```

def __init__(self, run_name):
    self._run_name = run_name
    self._run_timestamp = time.strftime("%Y%m%d-%H%M%S")

def write(self, obj: str, unique_id="") -> str:
    filename = self.generate_filename(unique_id=unique_id, suffix=".txt")
    with open(filename, "w") as f:
        f.write(obj)
    return filename

def write_csv(self, container: List[Union[Match, HashedFile]],
unique_id=""):
    filename = self.generate_filename(unique_id=unique_id, suffix=".csv")
    with open(filename, "w", newline="") as csvfile:

```

```

try:
    fieldnames = [column for column in container[0].__dict__.keys()]
except IndexError:
    # when container is empty
    return filename

writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
writer.writeheader()
for item in container:
    writer.writerow(item.__dict__)
return filename

def compress(self, container: List[str], unique_id=""):
    filename = self.generate_filename(unique_id=unique_id, suffix=".zip")
    with ZipFile(filename, "w") as zipfile:
        for file in container:
            zipfile.write(file)
    return filename

def generate_filename(self, unique_id="", suffix="") -> str:
    return f"{self._run_name}-{self._run_timestamp}-{unique_id}{suffix}"

@staticmethod
def inappropriate_format(file: str) -> bool:
    """
    should check if file is executable binary
    if it is return True, else return False
    """
    return Path(file).suffix in [
        ".zip",
        ".tar",
        ".gzip",
        ".7z",
        ".mp3",
        ".mp4",
        ".avi",
        ".jpg",
        ".png",
        ".gif",
    ]

@staticmethod
def resolve_path(

```



```

        root,
    ) -> Path.absolute:
        current = Path(root)
        current.expanduser()
        return current.absolute()

class Controller:
    def __init__(
        self,
        root=".",
        run_name=None,
        target_wallet=None,
        verbose: bool = False,
        silent: bool = False,
        quick: bool = False,
    ):
        self.root: str = root
        self._found_patterns: List[Match] = []
        self.run_name: str = run_name
        self.hashd_files: List[HashedFile] = []
        self.file_operator = FileOperator(run_name)
        self.processor = Processor()
        self.verbose = verbose
        self.silent = silent
        self.quick = quick
        self.target_wallet = target_wallet
        self.default_wallets_paths = {
            "exodus": "~/.config/Exodus",
            "electrum": "~/.electrum",
        }

    def specific_wallet_check(self):
        path = (
            Path(self.default_wallets_paths[self.target_wallet]).expanduser().absolute()
        )
        if not path.exists():
            return False
        return True

    def main(self) -> None:
        process_snapshot = self.processor.get_running_processes()
        command_history = self.processor.examine_command_history()

```

```

        indication =
self.processor.examine_process_snapshot(process_snapshot)

        explored_files: int = 0
        current_absolute = self.file_operator.resolve_path(self.root)
        for root, _, files in os.walk(current_absolute):
            for file in files:
                abs_path = root + "/" + file
                if self.quick and
self.file_operator.inappropriate_format(abs_path):
                    continue
                self.search_for_pattern(abs_path)
                explored_files += 1

        files = [match.file for match in self._found_patterns]
        total_patterns = [match.hit for match in self._found_patterns]
        regex_type = [match.type for match in self._found_patterns]

        files = list(set(files))
        for file in files:
            fingerprint = self._touch_sha256(Path(file))
            self.hashd_files.append(HashedFile(file, fingerprint))

        regex_type_count = Counter(regex_type).most_common(3)

        reports = None
        if self.silent:
            pass
        else:
            reports = [
                self.file_operator.write(
                    process_snapshot.stdout.decode("utf-8"),
                    unique_id="process_snapshot",
                ),
                self.file_operator.write_csv(
                    unique_id="hashd_files", container=self.hashd_files
                ),
                self.file_operator.write_csv(
                    unique_id="artefacts", container=self._found_patterns
                ),
                self.file_operator.compress(container=files),
            ]

        self.print_summary(

```

```

files=files,
indication=indication,
command_history=command_history,
reports=reports,
total_patterns=total_patterns,
machine_info=self.processor.get_host_information(),
target_wallet=self.target_wallet,
explored_files=explored_files,
regex_type_count=regex_type_count,
)

```

```

def add_match(self, match: Match) -> None:
    self._found_patterns.append(match)

```

```

def search_for_pattern(self, file: str, mode="rb"):
    counter = 0
    try:
        with open(file, mode) as f:
            for i, line in enumerate(f):
                for key in PATTERNS.keys():
                    match = re.search(PATTERNS[key], line)
                    if match:
                        self.add_match(
                            Match(file, match.group().decode("utf-8"), key)
                        )
                        counter += 1
                    if self.verbose:
                        print(f"Match found in file: {file}")
    except (FileNotFoundError, OSError):
        pass # file is probably a broken symbolic link or a network socket
    print(f"Exploring file: {file} [{counter} matches]")

```

```

def _touch_sha256(self, file: Path) -> str:
    # solution obtained from the following link:
    # https://stackoverflow.com/questions/22058048/hashing-a-file-in-
python
    h = hashlib.sha256()
    with open(file, "rb") as bin_file:
        while True:
            chunk = bin_file.read(h.block_size)
            if not chunk:
                break
            h.update(chunk)
    return h.hexdigest()

```

```

def print_summary(self, **kwargs):

    print("\n*** RUN SUMMARY ***\n")

    t = time.localtime()
    current_time = time.strftime("%H:%M:%S", t)
    print(f"Test executed at: {current_time}")

    print(f"System information: {kwargs['machine_info']}\n")

    print(f"Total number of explored files: {kwargs['explored_files']}")
    print(f"Files containing bitcoin related patterns: {len(kwargs['files'])}")
    print(f"Total number of found patterns:
{len(kwargs['total_patterns'])}")
    print(f"Total number of unique patterns:
{len(set(kwargs['total_patterns']))}")
    print(f"Wallet/bitcoin processes running: {kwargs['indication']}")
    print(f"Wallet/bitcoin command used: {kwargs['command_history']}")

    print(f"Three most common types of addresses:")
    for regex, count in kwargs["regex_type_count"]:
        print(f"{regex}: {count} hits")
    print()

    if kwargs["target_wallet"]:
        print(
            f"{kwargs['target_wallet']} wallet exists:
{self.specific_wallet_check()}"
        )

    if kwargs["reports"]:
        print(f"Results saved to: {kwargs['reports']}")

    print(
        "\nTIP: To save this output to a file, use output redirection (e.g.
python3 main.py --name test -q > quick_scan.txt)"
    )
    print("\n*** SUMMARY END ***\n")

def main():
    parser = argparse.ArgumentParser(description="Look for bitcoin
artifacts")

```

```

group_wallets = parser.add_mutually_exclusive_group()
parser.add_argument("--name", "-n", help="Name of the run",
required=True)
parser.add_argument(
    "--directory",
    "-d",
    help="Root directory to start the scan from. If not specified, then
current directory is the root directory",
    default=".",
)
group_wallets.add_argument(
    "--exodus",
    help="Search only for Exodus related files.",
    action="store_true",
)
group_wallets.add_argument(
    "--electrum",
    help="Search only for Electrum related files.",
    action="store_true",
)
parser.add_argument(
    "--verbose", "-v", help="Verbosely list files processed.",
action="store_true"
)

parser.add_argument(
    "--silent", "-s", help="Do not create any output files",
action="store_true"
)

parser.add_argument(
    "--quick",
    "-q",
    help="Do not scan files ending with the following suffixes: "
    "[.zip, .tar, .gzip, .7z, .mp3, .mp4, .avi, .jpg, .png, .gif]",
    action="store_true",
)

args = parser.parse_args()

if args.exodus:
    target_wallet = "exodus"
elif args.electrum:
    target_wallet = "electrum"

```

```

else:
    target_wallet = None

c = Controller(
    root=args.directory,
    run_name=args.name,
    target_wallet=target_wallet,
    verbose=args.verbose,
    silent=args.silent,
    quick=args.quick,
)
c.main()

if __name__ == "__main__":
    main()

regular_expressions.py
EXODUS = b"[eE]xodus"
ELECTRUM = b"[eE]lectrum"

PATTERNS = {
    "BITCOIN_P2SH": b"^[a-km-zA-HJ-NP-Z1-9]{24,33}", #
    multisignature,
    "BITCOIN_BECH32": b"bc1[a-zA-HJ-NP-Z0-9]{25,59}",
    "BITCOIN_LEGACY": b"[13][a-km-zA-HJ-NP-Z1-9]{25,34}",
    "EXTENDED_PUBLIC_KEY": b"xpub[a-km-zA-HJ-NP-Z1-9]{107,108}",
    "WIF_PRIVATE_KEY": b"5[a-km-zA-HJ-NP-Z1-9]{50}",
    "WIF_COMPRESSED_PRIVATE_KEY": b"[KL][a-km-zA-HJ-NP-Z1-9]{51}",
    "ENCRYPTED_PRIVATE_KEY": b"6P[a-km-zA-HJ-NP-Z1-9]{56}",
    "EXTENDED_PRIVATE_KEY": b"xprv[a-km-zA-HJ-NP-Z1-9]{107,108}",
    "MINI_PRIVATE_KEY": b"S[a-km-zA-HJ-NP-Z1-9]{29}",
}

```