

Received September 19, 2019, accepted October 10, 2019, date of publication October 21, 2019, date of current version November 12, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2948774

An Automated Live Forensic and Postmortem Analysis Tool for Bitcoin on Windows Systems

STEPHAN ZOLLNER¹, KIM-KWANG RAYMOND CHOO² , (Senior Member, IEEE), AND NHIEN-AN LE-KHAC³, (Member, IEEE)

¹Bavarian Police Institute for Further Education, Training Institute of the Bavarian Police, 83404 Ainring, Germany

²Department of Information Systems and Cyber Security, University of Texas at San Antonio, San Antonio, TX 78249-0631, USA

³School of Computer Science, University College Dublin, Dublin 4, D04 V1W8 Ireland

Corresponding author: Kim-Kwang Raymond Choo (raymond.choo@fulbrightmail.org)

ABSTRACT Bitcoin is popular not only with consumers, but also with cybercriminals (e.g., in ransomware and online extortion, and commercial online child exploitation). Given the potential of Bitcoin to be involved in a criminal investigation, the need to have an up-to-date and in-depth understanding on the forensic acquisition and analysis of Bitcoins is crucial. However, there has been limited forensic research of Bitcoin in the literature. The general focus of existing research is on postmortem analysis of specific locations (e.g. wallets on mobile devices), rather than a forensic approach that combines live data forensics and postmortem analysis to facilitate the identification, acquisition, and analysis of forensic traces relating to the use of Bitcoins on a system. Hence, the latter is the focus of this paper where we present an open source tool for live forensic and postmortem analysing automatically. Using this open source tool, we describe a list of target artifacts that can be obtained from a forensic investigation of popular Bitcoin clients and Web Wallets on different web browsers installed on Windows 7 and Windows 10 platforms.

INDEX TERMS Bitcoin forensics, bitcoin web wallet, cryptocurrency forensics, automated forensic process, digital forensics.

I. INTRODUCTION

Cryptocurrencies, such as Bitcoins, are becoming increasingly popular. Bitcoins can be used relatively anonymously and it can be challenging, or at times impossible, to determine the real identity of the owner of a Bitcoin address. Moreover, Bitcoin transactions are relatively cheaper and faster than a typical banking transaction. Hence, it is not surprising that Bitcoins are also been used by criminals in their illegal activities (e.g. paying for illicit drugs, and receiving payments in online extortion and ransomware cases). For example as recent as October 2019, hundreds of individuals were charged in the takedown of the largest, global darknet child abuse/exploitation material website, which was allegedly funded by Bitcoin.¹

The associate editor coordinating the review of this manuscript and approving it for publication was Qing Yang .

¹<https://www.justice.gov/opa/pr/south-korean-national-and-hundreds-others-charged-worldwide-takedown-largest-darknet-child> and <https://www.wired.com/story/dark-web-welcome-to-video-takedown-bitcoin/>, last accessed October 18, 2019.

Despite Bitcoins and other cryptocurrencies such as Litecoin, Namecoin, Freicoin and Peercoin and Monero being a potential source of evidence, there has been limited research on cryptocurrency forensics. In a related research [3], the authors described how one can conduct live forensics of a Bitcoin wallet on the Windows 7 platform. However, there is a wide range of Bitcoin wallet types, such as client wallets (also known as mobile applications - apps), web wallets, in-browser wallets, and hardware wallets, as well as varying configurations. In addition, there is a lack of approaches on Bitcoin forensics in literature such as the analysis of different formats of keys and addresses, the combination of live and post-mortem forensics, analysing the file signatures of Bitcoin relevant files. Moreover, the analysis of the most commonly used Bitcoin clients, to find more artifacts on the file system and in the registry, is also necessary.

Hence, in this paper, we present a forensic approach that combines both live and postmortem analysis to locate as many traces of Bitcoins as possible and eventually, the wallet's keys. We first describe a list of relevant target

artifacts, such as the Bitcoin keys and address, mnemonic codes, and prefetch files. We then demonstrate how our proposed forensic approach can be used to identify the fund controller and locate traces of Bitcoins. We evaluate our approach using four popular web browsers, namely: Google Chrome, Mozilla Firefox, Microsoft Internet Explorer and Microsoft Edge, as well as using the TOR browser and running the four browsers in their private modes. Our approach is also designed to facilitate searches of all located artifacts automatically, and for both live data and postmortem forensics on Windows 7 and Windows 10 platforms.

The rest of paper is organized as follows. Section II presents the extant literature. We describe the potential forensic artifacts relating to Bitcoin forensics and our forensic approach in Sections III and IV, respectively. We implement our approach in Section V, and discuss our findings in Section VI. Finally, we conclude this paper in the last section.

II. RELATED WORK

A tutorial on Bitcoin forensics was presented in [5], focusing only on the background knowledge on Bitcoins such as the encryption techniques applied to Bitcoin keys, and the structure of Bitcoin block in the blockchain, Bitcoin address, and Bitcoin wallet.

There are not many tools that support cryptocurrency forensics. For example, while Internet Evidence Finder (IEF) [4] since version 6.1 can be used to parse wallet files and client log files, and Bitcoin clients' on-disk resources, IEF supports only Bitcoin Core client forensics at the time of this research. The authors in [6] presented an open source program "BTCscan" based on Python, but the program only automates the extraction of Base58Check encoded strings in ANSI or Unicode. In [7], the author analysed Litecoin and Darkcoin installed on iOS and Android devices, and was able to recover relevant metadata, such as installation date and time stamps and usage indicators. For the iOS bitWallet app, the author also recovered a private key in plaintext. However, the author focused only on human-readable data and not binary data. In [8], the authors focused on forensic artifact acquisition of seven Android cryptocurrency wallet apps. In another work [9], Bitcoin-QT and Multibit apps on a Windows device were investigated. A list of application files of forensic value was obtained. The string-based (human-readable) data was examined, and analysis of the memory snapshots was performed using keyword search in EnCase. While the latter (a form of unstructured memory analysis) yielded many hits, there is no detailed analysis of the memory location. In addition, the data format and its implications were not presented.

The authors in [3] presented a memory forensic analysis of two popular Bitcoin clients, namely: Bitcoin Core and Electrum. They demonstrated that Bitcoin forensic artifacts can be extracted from memory by scanning the process memory or by searching fixed patterns with regular expressions (e.g. Bitcoin addresses or file paths). They also pointed out that most data found in memory are also available in

application and wallet files on disk, with a few exceptions. As such, process memory analysis is beneficial to a forensic investigation, particularly when application and wallet files are not available. The authors focused on both string values and binary-formatted values stored as string or binary. However, their research was limited to memory forensic analysis on a specific operating system. The authors of [10] also described a memory forensic investigation for Bitcoin, CryptoNote and Monero. They analysed the memory dump to identify the transaction and the wallet of these cryptocurrencies. However, they did not look at postmortem approaches for the forensic acquisition of cryptocurrencies.

III. TARGET ARTIFACTS

A. BITCOIN KEYS AND ADDRESSES

Bitcoin private keys, public keys and addresses can exist in different formats, although most Bitcoin keys and addresses are Base58 encoded, perhaps with the exception of the mini private key that uses only the Base58 alphabet. The length of the Base58Check encoded keys or addresses depends on the input and especially on the prefix, which are the most significant bits in big-endian. To search the system, it can be helpful if the keys or addresses are Base58Check encoded, because they contain only letters and numbers from the Base58 alphabet, and the integrated checksum ease the validation of the string and filtering of false positives.

1) PRIVATE KEYS

The private key in hexadecimal format consists of 64 hexadecimal digits. This is not very useful for the search on a system, because everything can be a private key and the result would be lots of false positives. For this reason, the private key in this format will not be considered in our paper. The private key in compressed hexadecimal format also consists of 64 hexadecimal digits and an additionally added suffix "_01_". The suffix would make it a little easier to find the private key on a system, but there are still 64 hexadecimal digits at the front of the suffix that cannot be verified. The result would also have too many false positives and will, therefore, not be considered in our paper too. The private key in WIF format consists of a 51 characters long string, starting with a prefix and is Base58Check encoded. This WIF private key starts with the number 5, which is 0×80 in hexadecimal. The total size of input bytes (needed for the Base58Check encoding) is 37 bytes, and can be calculated as follows: 1 byte for the prefix 0×80 , 32 bytes input from the private hex key (64 hexadecimal digits), and a checksum which results from the first 4 bytes of the SHA-256 cryptographic hash function. The private key in WIF compressed is similar to the normal WIF private key, with the exception of having a suffix of 01.

The encrypted private key consists of a 58 characters long string, starting with a prefix 6P and is also Base58Check encoded. According to BIP38, the total size of input bytes (needed for the Base58 encoding) is 43 bytes and can be

calculated as follows: 2 bytes for the prefix 0×0142 , 1 flagbyte, 4 bytes salt, 32 bytes for the encrypted payload and a checksum which results from the first 4 bytes of the SHA-256 cryptographic hash function. The extended private key consists of a up to 112 characters long string, starting with a prefix `xprv` that is $0 \times 0488ADE4$ in hexadecimal and is also Base58Check encoded. The total size of input bytes is 82 bytes and can be calculated as follows: 4 bytes for the prefix $0 \times 0488ADE4$, 1 byte depth, 4 bytes fingerprint, 4 bytes child number, 32 bytes chain code, 33 bytes for the data and a checksum which results from the first 4 bytes of the SHA-256 cryptographic hash function.

The mini private key consists of a 30 characters or 22 characters long string, starting with a prefix, which is the character `S` and using the Base58 alphabet. According to `bcwiki:minipk` [11], the mini private key is only valid if the following steps are completed and the results achieved: (1) add a question mark to the end of the string; and (2) then, the SHA-256 cryptographic hash of the entire string must be calculated. If the first byte of the result is `00`, then the key is valid.

2) PUBLIC KEYS

The compressed public keys consist of 66 hexadecimal digits, with the two different prefixes `"02"` and `"03"`. The challenge is relatively similar to that of the private keys in hexadecimal format. A search on a system with this structure would result in lots of false positives. For this reason, the public key in this format will not be considered in our paper. The uncompressed public key is similar to the compressed public key, except that the prefix `"04"` is used and the length is 130 instead of 66 hexadecimal digits. In fact, a search would also return too many false positives and because of this, the uncompressed public key will not be considered in our paper either. The extended public key consists of up to 112 character-long string, starting with a prefix `xpub`. This is $0 \times 0488B21E$ in hexadecimal and is also Base58Check encoded. The computation approach is similar to that of the extended private key.

3) BITCOIN- AND VANITY ADDRESS

The "normal" Bitcoin address and the vanity address are similar in their structure. They consist of 25 to 34 characters, starting with the same prefix and they are also Base58Check encoded. Both addresses start with the number 1 (0×00 in hexadecimal). The total size of input bytes is 25 bytes and can be calculated as follows: 1 byte for the prefix 0×00 , 20 bytes input from the cryptographic hash function RIPEMD-160 and a checksum which results from the first 4 bytes of the SHA-256 cryptographic hash function.

4) MULTI-SIGNATURE ADDRESS

The multi-signature address is similar to the other two addresses above, except for the different prefixes. The addresses start with the number 3 (0×05 in hexadecimal). The length is between 25 and 34 alphanumeric characters.

The total size of input bytes can be calculated as follows: 1 byte for the prefix 0×05 , 20 bytes input from the cryptographic hash function RIPEMD-160 and a checksum which results from the first 4 bytes of the SHA-256 cryptographic hash function.

B. MNEMONIC CODE WORDS, WALLET ID AND EMAIL ADDRESS

1) MNEMONIC CODE WORDS

Mnemonic code words can be used to generate deterministic wallets. How to generate the mnemonic code words and the seed is specified in BIP39. Furthermore, there is a recommended list of words in several languages available for download. Mnemonic codes are typically 12, 18 or 24 words long. However, some wallets allow to add words manually until a maximum of 24 words. Moreover, only words in lower case and between three to eight characters are used. The words must be encoded in UTF-8, which for English words is 8 bits or 1 byte.

2) WALLET ID

In some web wallets, a wallet ID is used as the username. Technically it can be split into five groups, each using lower case letters and numbers, separated by a hyphen `"-"`. The first group contains eight alphanumeric characters, each of the next three groups contains four alphanumeric characters and the last group contains 12 alphanumeric characters.

3) EMAIL

Similar to wallet IDs, an email address is used in some web wallets as the username. Typically, an email can be split into three logical groups. The first group can be upper and lower case, numbers and the special signs `"_"`, `"."`, `"+"` or `"-"` and a minimum of one character followed by a `@` symbol. The second group consists of a minimum of one character, which can be upper and lower case as well as numbers followed by a `"."`. The third and last group consists of a minimum of two and a maximum of four signs which can be upper and lower case, numbers or special signs like `"_"` or `"."`.

C. BITCOIN CLIENTS AND WEB WALLETS

During the analysis of **Bitcoin-clients** for traces on the file system, one should note that the registry and the processes are started by the client. From the analysis of clients such as Bitcoin Core, Armory, Electrum (Windows Installer), Electrum Portable, mSIGNA, Bitcoin Knots, Bither, MultiBit HD and Copay/BitPay, the obtained traces include added files and folders on the file system and added registry keys. Moreover, artifacts left on the file system or in the registry after uninstalling the software should also be analysed.

An alternative to Bitcoin-clients is **web wallets**. Such wallets (e.g. Coin.Space, BitGo, Green Address, Coinapult, Coinbase, Xapo and Blockchain.info) can be used in different browsers with different modes. To find traces of bitcoins in

browser artifacts, the different browser modes and the way they store their artifacts should be analysed.

D. FILE SIGNATURES AND KEYWORDS

For bitcoin related files, there are also some available **file signatures**. The list is limited and all the available file signatures are related to Bitcoin Core and Multibit. Such signatures can be included in a binary search on the system, to expedite the search. Another common way of searching and finding traces of bitcoins is through the lists of **keywords**. These lists can include keywords from other artifacts or common keywords that are related to cryptocurrencies. These lists are “living” lists, which means that they will never be finalized and will continue to be updated.

E. OTHER ARTIFACTS

Prefetch files are not Bitcoin-specific, but there are two key reasons for analysing these files. First, the analysis of such files could facilitate the identification of the Bitcoin client previously executed on the system and when the last execution took place. Second, it also allows us to determine which browser was used by the user and how artifacts of the web wallets can be analysed. Optional artifacts include the **RAM**, the **pagefile.sys**, and the **hiberfil.sys**, since it is not always possible to acquire these three artifacts. There are special forensic tools to capture and process the data stored in the RAM, the pagefile.sys or the hiberfil.sys. With an in-depth analysis of these artifacts, other useful evidence may be found.

IV. PROPOSED APPROACH

The objective of our research is to locate and extract important Bitcoin artifacts, such as private/public key, addresses or other traces of Bitcoins in a system. This information can be used for asset forfeiture (e.g. in Proceeds of Crime-type legislation) or can be important for forensic investigations. To achieve this goal, our forensic process includes two parts, namely: control of the wallet/account and finding traces of Bitcoins.

A. CONTROL OF THE WALLET/ACCOUNT

There are different options to take control of the wallet/account. The best option is to find and extract the private key. There are several ways how private keys can be hidden and stored, but once the private key is found, the key can be imported in a self controlled wallet and the Bitcoins can be deprived from the owner of the wallet. There is one exception, private keys which are implemented in BIP38 format² cannot be imported into a wallet without knowing the additional password or passphrase. However, this could be circumvented using the approach described by the authors in [2].

The second option is to find a wallet on a system that contains the private key. If the wallet is not encrypted, then the key will be stored locally and can be found and extracted

like what was discussed in the preceding paragraph. If the wallet is encrypted, then an additional password is needed. Hence, a brute-force attack is required unless the wallet in question also suffers from the same vulnerability described in [2]. We combine both extracting private keys and finding the wallet in our approach.

B. FINDING TRACES OF BITCOINS

In most cases, finding traces of Bitcoins is a complex procedure and requires an in-depth deep knowledge of the system and where traces can be found. Searching manually is only partly possible because of the number and variety of devices (desktop or laptop computers, mobile phones, tablets or flash drives) as well as their operating systems. In addition, there are a large number of different formats of Bitcoin private and public keys, and addresses can appear different and can be built using different algorithms. Indeed, there are also different types of wallets that can be used for Bitcoin transactions. Most of them are Bitcoin-clients or web wallets. The latter can be used in different browsers with different modes (normal or private). Each one of the four popular browsers provides two modes, which are different in the way they store data. In the normal mode, browsers record a wealth of information like the history, cookies, bookmarks, temporary internet files and typed URLs. For example, Chrome provides a normal and an Incognito mode, Mozilla Firefox has a normal mode and a private browsing mode, and Microsoft Internet Explorer and Microsoft Edge provide a normal mode and an InPrivate mode. Additionally, TOR browser (based on the Firefox browser) is another popular browser.

In order to find all relevant evidence related to Bitcoin keys, Bitcoin-client, Web wallets, file signatures, etc. in our approach, we use both live data forensics (LDF) and post-mortem analysis. For LDF to find traces of bitcoins, the following order is recommended in order to achieve the best results:

- 1) Acquire the contents of the physical memory (RAM). The information inside the RAM is essential, for example artifacts like web wallets which are in private or incognito mode or the artifacts from the TOR browser. Moreover, passwords or other Bitcoin-related artifacts, such as keys, may be found. In addition, acquire the contents of files hiberfil.sys and pagefile.sys, whose content can be similar to the RAM.
- 2) Search for file meta-data on the hard disk. This information is less volatile than the earlier two pieces of information (see items a and b above), but meta-data can be changed on a running system and also if the system is powered down or rebooted. For example, timestamp is meta-data and can be extremely important in court proceedings.
- 3) Search for file content, removable media and backup media for the artifacts described in this paper. These points, for example, also include the encryption of Bitcoin keys.

²Bitcoin Improvement Proposals.

While performing postmortem forensics, in our approach, we can search for Bitcoin addresses and private or public keys on the system. Furthermore, it is possible to search for traces of Bitcoin-clients on the file system or in the registry. It is also possible to search for file signatures or browser artifacts of web wallets, assuming that a private mode of a browser or a TOR browser was not used. Moreover, we can also search for relevant keywords on the system. On some operating systems such as Windows, it is not certain that the memory is completely lost even if a postmortem forensic examination is conducted. If the system has been put into the hibernation mode, the file hiberfil.sys (that contains a complete memory dump) is written to the drive. Moreover parts of the memory can be found in the file pagefile.sys.

C. FORENSIC PROGRAM TO AUTOMATE THE SEARCH

Not only the target artifacts are essential, a program to automate the search for all artifacts and tasks which are described are important too. The program is not an all-in-one solution for all possible searches in LDF or postmortem forensics. There are a couple of requirements which must be observed when developing the software. These requirements include:

- The software must include possibilities to use it **live** on a running system and for **postmortem** forensics.
- All worked out artifacts must be handled by the software in **modules** and **functions**.
- The **search** function must be divided into at least two parts, a quick search and a detail search.
- The software must be **portable** and executable without dependencies.
- The **handling** of the software must be easy so that everyone can use it by reading the manual and without the requirement of specific training.
- The created **report** must observe forensic standards and also be useable as proof in court proceedings.
- The code must be well **documented**. This is important for forensic work because the system changes on a suspects machine should be as little as possible and if changes are made, they need to be well known. Furthermore, the code is needed for error handling.

V. IMPLEMENTATION

Our approach is implemented as a software tool using Python version 3.5.3. The naming convention is closely based on the web page python:naming [12]. To develop this software tool, the described artifacts must be mapped in the software and supplemented with common points. This results in a rough division in the following modules which can be developed separately and integrated in a main module:

- **Main**. This is the main module of the software for a centralised processing of all tasks and to control the behaviour of the software.
- **Global**. This is a module for global functions and variables which can be easily integrated and used in every other module.

- **Prefetch**. This module is responsible for analysing the prefetch files, which can be used to find bitcoin-clients or help to identify which browsers are used.
- **Search**. This module is divided into three subcategories, the regular expression search, the keyword search and the file signature search.
- **Browsers**. This is the main module for analysing browser artifacts. The behaviour depends on if a database or files are used to save data behind a certain browser. Figure 1 gives an overview of the different browsers and the way which they save their data.

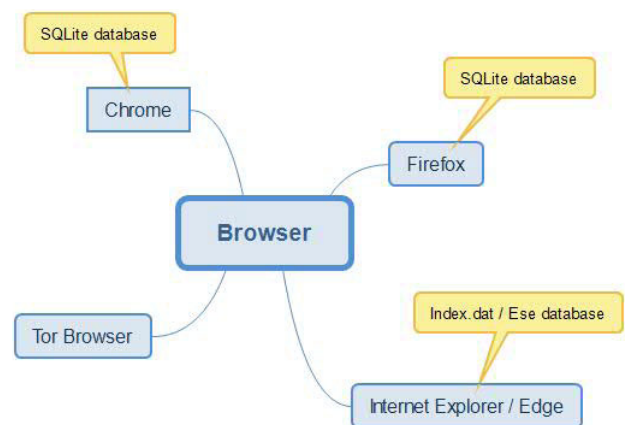


FIGURE 1. Browsers and how data is saved: A simplified overview.

- **Clients**. This is the main module for analysing the certain client artifacts on the file system or in the registry. Moreover, it checks the running processes during LDF.
- **Memory**. This module is responsible to acquire the RAM, the pagefile.sys and the hiberfil.sys on a running system and the postmortem analysis.
- **Logging**. This is the main module for logging different information. This includes system information, errors and certain steps like system changes or recording timestamps of approach steps for the use in court proceedings.

Our software tool is called **WinBAS** which stands for **Windows Bitcoin Artifact Scanner**. This tool is available under the following link: <https://github.com/Guichave/WinBAS>. In this paper, only the most important parts of the source code are described and analysed. In general, most of the methods and functions for the export to files, print to the command line, the error handling and for the internal program handling will not be described in this paper. The functionalities of main components are described as follows:

A. "WINBAS"

The main module WinBAS.py is the control center of the software. It includes the following main functions:

- Checking the environment including administrator privilege, existing of web browsers, required arguments, etc.
- Setting relevant environment variables.

- Creating the working environment e.g. directories for results and data.
- Exploring the file systems.
- Creating MD5 hashes for all created files.
- Exporting results.

For some tasks in the program, WinBAS must be started as administrator. These tasks may include, but are not limited to: acquiring the RAM, the pagefile.sys or the prefetch files. In addition, the Windows environment variables are very essential. This has two reasons: The first one is to receive information about the foreign system, for example if it is a 32 bit or an 64 bit architecture. The second reason is to map common parts of the detected artifacts to the concrete environment, for example <user name> refers to the user which is currently logged in on the running system.

Besides, due to the fact that this tool only works with Microsoft Windows, browser artifacts from Internet Explorer and Edge will always be checked. In the case of Mozilla Firefox and Google Chrome, there are two possibilities to determine if the browser is installed on the system. The first possibility is to look at the typical path on the file system. This can be:

Google Chrome

- C:\Program Files (x86)\Google\Chrome\Application
- C:\Program Files\Google\Chrome\Application
- C:\<user name>\AppData\Local\Google\Chrome

Mozilla Firefox

- C:\Program Files (x86)\Mozilla Firefox
- C:\Program Files\Mozilla Firefox
- C:\<user name>\AppData\Roaming\Mozilla\Firefox
- C:\<user name>\AppData\Local\Mozilla\Firefox

The second possibility is to check the prefetch files for the executed browsers. If either a prefetch file of a certain browser exists or one of the path, the browser will be checked for artifacts.

B. "LOG"

Logging is a small but very essential module for two reasons. The first is that it documents every step during the execution of WinBAS, for example when a registry key is changed or a folder is created. Especially in the field of LDF this is very important because it is a foreign system and the results could be needed in court proceedings. The second reason is to document every error which occurs during the execution of the software. This is very important for increasing the software quality or reconstructing the error to solve the problem. To realise the logging, there is only one function "global_log(name, file_name)" in the module log.py. The first parameter contains the name of the current module from which the log entry comes from. The second parameter is the name of the current log file.

C. "PREFETCH"

To acquire the Windows prefetch files, the tool Win-PrefetchView from NirSoft (<https://www.nirsoft.net>) is

TABLE 1. Python regular expressions (Standard Encoding).

Name	Standard encoding (8 bit)
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}'
Extended public key	b'xpub[a-km-zA-HJ-NP-Z1-9]{107,108}'
Bitcoin and vanity address	b'1[a-km-zA-HJ-NP-Z1-9]{24,33}'
Multi-signature address	b'3[a-km-zA-HJ-NP-Z1-9]{24,33}'

started as a subprocess. To analyse the output of the acquired text file, a regular expression is used to extract the name from the prefetch file. The standard module "mmap" is used to achieve a higher performance than when using a typical regular expression search. This module uses the virtual memory of the OS to access the data on the file system directly, instead of using I/O functions.

D. "FSSEARCH"

The Python module fssearch.py performs different searches on the file system which can be split into three main categories: search with regular expressions, search with file signatures and search with lists of keywords. Furthermore, some general module functions are necessary to coordinate the search and implement specific tasks.

TABLE 2. Python regular expressions (Unicode Encoding).

Name	Unicode (16 bit)
WIF private key	b'5\x00([a-km-zA-HJ-NP-Z1-9]\x00){50}'
WIF comp. private key	b'[KL]\x00([a-km-zA-HJ-NP-Z1-9]\x00){51}'
Encrypted private key	b'6\x00P\x00([a-km-zA-HJ-NP-Z1-9]\x00){56}'
Extended private key	b'x\x00p\x00r\x00v\x00([a-km-zA-HJ-NP-Z1-9]\x00){107,108}'
Mini private key	b'S\x00([a-km-zA-HJ-NP-Z1-9]\x00){29}'
Extended public key	b'x\x00p\x00u\x00b\x00([a-km-zA-HJ-NP-Z1-9]\x00){107,108}'
Bitcoin and vanity address	b'1\x00([a-km-zA-HJ-NP-Z1-9]\x00){24,33}'
Multi-signature address	b'3\x00([a-km-zA-HJ-NP-Z1-9]\x00){24,33}'

1) REGULAR EXPRESSIONS

This implements the transformation of the described structures above into regular expressions. Furthermore, it coordinates the regular expression search on the file system and validates the hits. Tables 2 and 2 show the regular expressions for standard and Unicode encoding, respectively.

config	28.07.2017 18:25	Dateiordner	
data	28.07.2017 18:28	Dateiordner	
logs	28.07.2017 18:28	Dateiordner	
results	28.07.2017 18:28	Dateiordner	
BITCOINRESULTS.txt	28.07.2017 18:19	Textdokument	1 KB
WinBAS.exe	28.07.2017 18:05	Anwendung	18.301 KB

FIGURE 2. WinBAS files and folders: An example.

The search with the regular expressions is similar to the prefetch file search. In addition, the Python standard module “mmap” is used to increase the performance of the software. The only difference to the prefetch file search is that two dictionaries are iterated with the regular expressions as keys and the associated byte size (needed for Base58Check) as values. The validation of the search hits is performed with Base58Check.

2) SIGNATURE

The known bitcoin file signatures are saved in a list. For every file, the first 8 bytes are read, converted into hexadecimal with the Python standard module “binascii” and compared with the list of searched file signatures. For the search with a given hexadecimal byte sequence it is important to know if the system stores the bytes in little-endian or in big-endian format.

3) KEYWORDS

The search with the list of keywords has two difficulties: Lots of different file types use their own specific file format, for example PDF, doc, docx and many more and the file encoding must be known and decoded to search with plain text keywords.

E. “CLIENTS”

This module searches for concrete path on the file system, keys in the registry and running processes. The search lists are administrated in the file Clients.ini.

F. “BROWSERS”

This module is split into three classes: InternetExplorerEdge, Chrome and Firefox. Moreover, some general module functions, which are shared by all classes, or functions which are responsible for the coordination of the three classes, are necessary. Table 3 gives an overview how browser artifacts acquired in this module.

TABLE 3. Browser artifacts: An overview.

	Chrome	Firefox	IE	Edge
History		BrowsingHistoryView.exe		
Bookmarks/Favorites	Text file	places.sqlite	favview.exe	ESEDatabaseView.exe
Cache	ChromeCacheView.exe	MozillaCacheView.exe		IECacheView.exe
Cookies	Cookies	cookies.sqlite		Text files
User data	ChromePass.exe	-		iepv.exe

To analyse the browser data, only two different functions are used which are similar. They use a regular expression to search for and compare the results of the search with the keyword list from the file Browser.ini and return the search hits.

G. “MEMORY”

To acquire the RAM and if available the pagefile.sys or the hiberfil.sys, the three external tools winpmem_1.6.0.exe16 and RawCopy.exe (32 bit architecture), RawCopy64.exe (64 bit architecture) from <https://hddguru.com/> are used. For all three artifacts there is an individual method to start a new subprocess to acquire the data.

H. REPRESENTATION OF THE RESULTS

After the execution of WinBAS, new files and folders are automatically created. The file BITCOINRESULTS.txt contains an overview of the found artifacts. Inside the folder results, the detailed results are listed. For each search step, a separate results file with the hits and the associated files (absolute path) is created. Figure 3 shows the listed files. Inside the folder data, all acquired artifacts are stored (without filtering for bitcoin artifacts). That can include the RAM dump or the copy of the pagefile.sys, the acquired data, for example the complete history of the browser Firefox. The folder FsSearch contains files which were hit with regular expressions and the folder IE_EdgeCookies contains all found cookies of the browsers IE or Edge. Figure 4 shows an example output.

BitcoinChrome.txt	28.07.2017 18:07	Textdokument	1 KB
BitcoinClients.txt	28.07.2017 18:07	Textdokument	1 KB
BitcoinFirefox.txt	28.07.2017 18:07	Textdokument	1 KB
BitcoinFsKeyword.txt	28.07.2017 18:19	Textdokument	97 KB
BitcoinFsRegex.txt	28.07.2017 18:19	Textdokument	9 KB
BitcoinFsSignature.txt	28.07.2017 18:19	Textdokument	1 KB
BitcoinIE_Edge.txt	28.07.2017 18:07	Textdokument	1 KB
BitcoinPrefetch.txt	28.07.2017 18:07	Textdokument	1 KB
HashesMd5.txt	28.07.2017 18:19	Textdokument	5 KB

FIGURE 3. WinBAS listings.

FsSearch	28.07.2017 18:28	Dateiordner	
IE_EdgeCookies	28.07.2017 18:28	Dateiordner	
ChromeCache.txt	28.07.2017 18:07	Textdokument	5 KB
ChromeCookies.txt	28.07.2017 18:07	Textdokument	1 KB
ChromeHistory.txt	28.07.2017 18:07	Textdokument	8 KB
ChromeUserData.txt	28.07.2017 18:07	Textdokument	0 KB
Dump.raw	28.07.2017 18:06	RAW-Datei	3.145.728 KB
FirefoxBookmarks.txt	28.07.2017 18:07	Textdokument	1 KB
FirefoxCache.txt	28.07.2017 18:07	Textdokument	26 KB
FirefoxCookies.txt	28.07.2017 18:07	Textdokument	1 KB
FirefoxHistory.txt	28.07.2017 18:07	Textdokument	9 KB
IE_EdgeCache.txt	28.07.2017 18:07	Textdokument	46 KB
IE_EdgeHistory.txt	28.07.2017 18:07	Textdokument	8 KB
IE_EdgeUserData.txt	28.07.2017 18:07	Textdokument	0 KB
IEFavorites.txt	28.07.2017 18:07	Textdokument	4 KB
pagefile.sys	28.07.2017 18:07	Systemdatei	3.145.208 KB
Prefetch.txt	28.07.2017 18:07	Textdokument	22 KB

FIGURE 4. Acquired data: An example.

VI. FORENSIC INVESTIGATION AND FINDINGS

In this section, we focus on the forensic examination of Bitcoin Clients and Bitcoin Web Wallets and their artifacts. To find traces of Bitcoins, several virtual machines (VMs) are used as a test environment. The installed operating systems (OSes) are Windows 7 (32 bit and 64 bit) and Windows 10 (64 bit, x64).

TABLE 4. Client analysis: An overview

	Armory	Multibit HD	Electrum (Windows Installer)	Electrum (Portable)	mSIGNA
Client type	Installer, 64 bit	Installer, 64 bit	Installer, 32 bit	Portable, 32 bit	Installer, 64 bit
Default Directory	C:\Program Files (x86)\Armory	C:\Program Files (x86)\MultiBit HD	32 bit: C:\Program Files\Electrum 64 bit: C:\Program Files (x86)\Electrum	-	C:\Program Files (x86)\mSIGNA
Process name	ArmoryQt.exe	multibit-hd.exe	electrum-2.8.3.exe	electrum-2.8.3-portable.exe	msigna.exe
64-bit specifics	All artifacts are x64 based	All artifacts are x64 based	-	-	All artifacts are x64 based
After uninstalling	Files in <user>2\Armory\ - armorycpplog.txt - armorylog.txt - ArmorySettings.txt	Files in <user>2\MultiBitHD\ - mbhd.yaml - mbhd-cacerts - logs\multibit-hd.log	Files in <user>2\Electrum\ - config - daemon - blockchain_headers - wallets\default_wallet	Files in C:\Users\<user name>\Desktop\electrum_data\ - blockchain_headers - config - daemon - \wallets\default_wallet	Files in <user>2\mSIGNA_bitcoin\ - debug.log

We use our software tool described above and this tool is also designed to allow us to perform **live** forensics on a running system, as well as **postmortem** forensics. The tool supports both quick and detail searches.

A. BITCOIN CLIENTS

To keep the system changes to a minimal, the network connection (including Internet) on the test environment is deactivated. For every installation or execution of a client, the suggested default locations and settings are accepted, even so they could have been changed. Because of this, some 64 bit installers use the directory C:\Program Files (x86).

Generally, after analysing all clients there are three main categories of artifacts that can be used for the search:

- 1) Common artifacts: Clients may leave similar artifacts in different environment, which can be included in the search. Example include executable files of a client like bitcoin-qt.exe for the client Bitcoin Core.
- 2) Specific artifacts: There are some specific artifacts for the different architectures of Windows 7 and also Windows 10 or the clients which are portable executable.
- 3) Artifacts after uninstalling or executing the client: Every client keeps traces of the installation process after uninstallation. Moreover, the portable executable of a client keeps traces too.

Windows 10 seems to create additional artifacts. For all installations, it creates artifacts which start with "C:\Program Files\OpenSSH\home". After comparing and inspecting the "home" part of the path, it is a link to the main directory of the users on the machine. So it must not be added separately to the search.

Most of the clients either use a 32 bit installer or a 64 bit installer. Only Bitcoin Core, Bitcoin Knots and Bither have an installer or executable for both architectures.

The artifacts found in different Bitcoin Clients are described in Table 4 and Table 5 with three categories: common artifacts, specific artifacts and artifacts after uninstallation.

B. BITCOIN WEB WALLETS

Web browsers are normally important sources of forensic investigation [14]. In terms of Bitcoin investigation, artifacts such as history, cookies, bookmarks, temporary Internet files and typed URLs can also be recovered from the analysis of Bitcoin web wallets, as explained below.

We register an account with each of the following web wallets: <https://blockchain.info/>, <https://www.bitgo.com/>, <https://coin.space/>, <https://greenaddress.it/>, <https://coinapult.com/>, <https://www.coinbase.com/>, and <https://xapo.com/>.

During the registration process, some wallets generate special artifacts that could be of forensic interest. For example, the wallet bitgo.com generates an additional file ("BitGo Keycard for My BitGo Wallet.pdf"), which must be stored locally. coin.space and greenaddress.it generate a passphrase with 12 to 24 mnemonic words. There is a high chance that the mnemonic words are also stored locally too.

For the first evaluation, a Windows 7 (32 bit, x86) test environment is used to examine browser artifacts. To validate the results and show the differences, the same evaluation is performed with a Windows 10 (64 bit, x64) environment.

1) ARTIFACTS OF BROWSERS IN NORMAL BROWSING MODE

a: MOZILLA FIREFOX

The URL of the Bitcoin web wallets being examined is presented in the table "moz_places" within the database places.sqlite. Additionally, the table "moz_bookmarks" within the same database is responsible for all bookmarks created in the browser. After opening the web pages and logging into the web wallets, for each wallet Firefox stores one or more cookies in the table cookies.sqlite. It's column "Domain/Host" contains the names which can be used for the search such as .app.xapo.com, .bitgo.com, .blockchain.info, and .coin.space. A range of artifacts are also located in the cache of Mozilla Firefox, which can be used for the search. Figure 5 shows an extract of the stored artifacts during

TABLE 5. Client analysis findings: An overview (cont.)

	BitPay/Copay	Bither	Bitcoin Core	Bitcoin Knots
Client type	Installer, 64 bit	Runnable executable, 32 bit and 64 bit	Installer, 32 bit and 64 bit	Installer, 32 bit and 64 bit
Default Directory	C:\Program Files (x86)\BitPay	-	C:\Program Files\Bitcoin	C:\Program Files\Bitcoin
Process name	BitPay.exe	Bither.exe	bitcoind-qt.exe	bitcoind-qt.exe
32-bit specifics	-	C:\Users\<user name>\AppData\Roaming\JWrapper-Bither\JWrapper-Windows32JRE-00051790352-complete\bin\Bither.exe	File system: C:\...\Bitcoin Core (32-bit).lnk Windows registry: HKLM\SOFTWARE\Bitcoin Core (32-bit) HKU\<SID>\Software\Bitcoin Core (32-bit)	File system: C:\...\Bitcoin Knots (32-bit).lnk Windows registry: HKLM\SOFTWARE\Bitcoin Knots HKU\<SID>\Software\Bitcoin Knots
64-bit specifics	All artifacts are x64 based	C:\Users\<user name>\AppData\Roaming\JWrapper-Bither\JWrapper-Windows64JRE-00051790437-complete\bin\Bither.exe	File system: C:\...\Bitcoin Core (64-bit).lnk Windows registry: HKLM\SOFTWARE\Bitcoin Core (64-bit) HKU\<SID>\Software\Bitcoin Core (64-bit)	File system: C:\...\Bitcoin Knots (64-bit).lnk Windows registry: HKLM\SOFTWARE\Bitcoin Knots HKU\<SID>\Software\Bitcoin Knots
After uninstaling	Files in C:\Users\<user name>\AppData\Local\bitpay\ User Data\Default\ - Cookies - History - Login Data	Files in <user>2\ - Bither\bither.db - Bither\address.db - Bither\log\bither.log - JWrapper-Bither\BitherWinLauncher.exe - Microsoft\Windows\Start Menu\Programs\Bither\Bither.lnk - JWrapper-Bither\12JWrapper-Windows32JRE-00051790352-complete\bin\Bither.exe - JWrapper-Bither\JWrapper-Windows64JRE-00051790437-complete\bin\Bither.exe	Files in <user>2\Bitcoin\ - wallet.dat - debug.log - db.log Windows registry: - HKLM\SOFTWARE\Bitcoin Core (32-bit) - HKLM\SOFTWARE\Bitcoin Core (64-bit)	Files in <user>2\Bitcoin\ - wallet.dat - debug.log - db.log Windows registry: - HKLM\SOFTWARE\Bitcoin Knots (32-bit) - HKLM\SOFTWARE\Bitcoin Knots (64-bit)

²<user>: C:\Users\<user name>\AppData\Roaming\

Filename	Content Type	URL	File Size
1497692203955		https://db.coin.space/cs303656871ee0c004...	41
1497692204005		https://db.coin.space/cs303656871ee0c004...	41
1497692401598	application/json	https://coinapult.com/wallet/balance?_id=14...	31
1497692401599	application/json	https://coinapult.com/wallet/balanceajax?...	159
14ePGaiwq/S...	application/json	https://api.blockchain.info/metadata/14eP...	341
14P65fuoirFLU...	application/json	https://api.blockchain.info/metadata/14P6...	290
1AZCcQTsbB...	application/json; ch...	https://live.coin.space/api/addr/1AZCcQT...	434
1DY6noqDwkj...	application/json	https://api.blockchain.info/metadata/1DY6...	43

FIGURE 5. Extract firefox cache files.

browsing of the web wallets. Useful information from the web wallet login process can be extracted, using the WebBrowserPassView tool from Nirsoft.³ Such information depends on the user behavior during the login process. For example, line 1 of Figure 6 shows the wallet ID and the password from the web wallet *blockchain.info*. Lines 2 and 3 show the 24 words passphrase and the PIN, which are needed to log in to the

³<https://www.nirsoft.net/>. The source code is available under Freeware License.

URL	Web Browser	User Name	Password
https://blockchain.info	Firefox 32+	076ba61a-5847-431f-8f6d-f93efdb38f50	zsch6vetae
https://greenaddress.it	Firefox 32+	vibrant between achieve donate key lottery ...	vibrant between achieve...
https://greenaddress.it	Firefox 32+		1807
https://www.coinbase.com	Firefox 32+	bitcointhesis@gmail.com	zsch6vetae

FIGURE 6. Extracted databases Nirsoft WebBrowserPassView for firefox.

web wallet *greenaddress.it*. Line 4 shows the email and the password needed to log in to *coinbase.com*.

b: CHROME

The column “url” of the table “urls” within the database *history* shows all requested web wallets like *coin.space* or *coinapult.com*, and the column “host_key” of the table *cookies* within the database “Cookies” shows different artifacts from the web wallets. Figures 7 and 8 show similar artifacts, like Mozilla Firefox before, which can be used in the same way to search for web wallets.

c: INTERNET EXPLORER

Internet Explorer until Version 9 uses files to store all artifacts. From Version 10, the artifacts are stored in a database

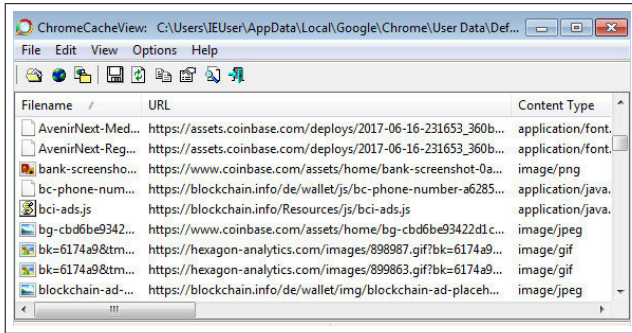


FIGURE 7. Extracted chrome cache.

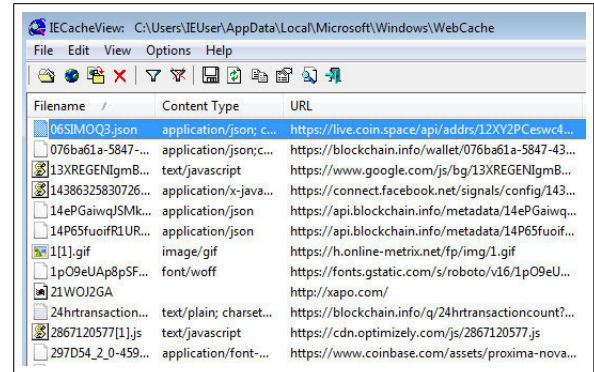


FIGURE 11. Extracted internet explorer cache.

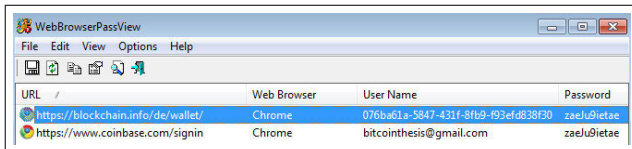


FIGURE 8. Extracted Nirsoft WebBroserPassView for chrome.

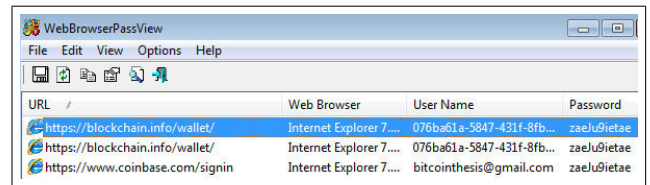


FIGURE 12. Extract Nirsoft WebBroserPassView for internet explorer.

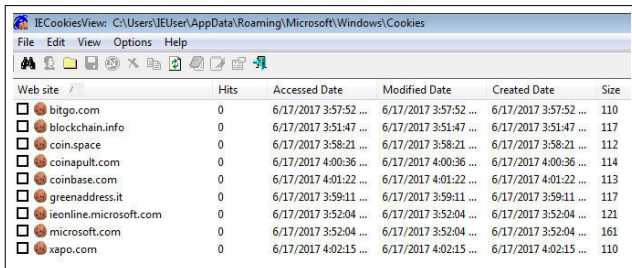


FIGURE 9. Extracted internet explorer cookies.

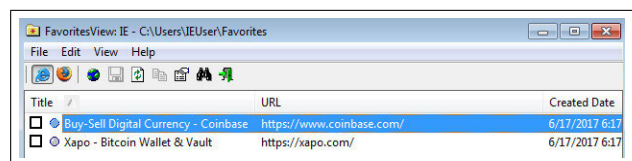


FIGURE 10. Extracted Windows 7 internet explorer favorites.

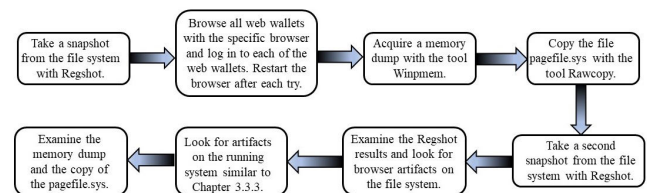


FIGURE 13. Procedure for analysing browser private or incognito modes.

2) ARTIFACTS OF BROWSERS IN INCOGNITO, INPRIVATE AND PRIVATE BROWSING MODES

To find traces of Bitcoin artifacts in the different private and incognito modes, the following procedure is used for every browser: To acquire the RAM and if available the pagefile.sys or the hiberfil.sys, the three external tools winpmem_1.6.0.exe⁴ and RawCopy.exe (32 bit architecture) and RawCopy64.exe (64 bit architecture)⁵ are used.

a: FIREFOX PRIVATE BROWSING MODE

Most of the web wallets do not work without allowing cookies. The web wallet coin.space does not work despite allowing cookies in private browsing mode and will not be considered for the test with Firefox private browsing mode. After closing the browser session, all cookies are found to have been deleted from the database. Additionally, there are no traces left in the browser history or the browser cache.

⁴<https://github.com/google/rekall/tree/master/tools/windows/winpmem>. The source code is available under the Apache License, Version 2.0.

⁵<https://hddguru.com/>. The source code is available under the Freeware License.

```

Rule: urls
Owner: Process svchost.exe Pid 1180
0x00731b76 63 00 6f 00 69 00 6e 00 62 00 61 00 73 00 65 00
0x00731b86 2e 00 63 00 6f 00 6d 00 00 00 90 be b1 1e 06 40
0x00731b96 00 14 00 00 00 00 00 00 00 01 00 04 00 09 00
Rule: urls
Owner: Process svchost.exe Pid 1180
0x0011f600 78 00 61 00 70 00 6f 00 2e 00 63 00 6f 00 6d 00
0x0011f610 00 00 00 00 26 00 00 00 00 00 26 00 00 00
0x0011f620 66 00 72 00 61 00 6d 00 65 00 77 00 6f 00 72 00

```

FIGURE 14. Volatility hits for firefox private browsing history.

```

Rule: advanced
Owner: Process vmtocled.exe Pid 2900
0x0075d295 7a 61 65 4a 75 39 69 65 74 61 65 00 00 00 00
0x0075d2a5 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0075d2b5 00 00 00 00 00 00 00 00 00 01 00 00 00 00 00
0x0075d2c5 00 00 00 00 00 01 00 72 65 62 65 6c 20 70 6f
0x0075d2d5 77 65 72 20 68 69 69 67 68 74 20 70 61 6e 64 61
0x0075d2e5 20 6d 6f 72 65 20 74 6f 6f 74 68 20 61 6c 63 6f
0x0075d2f5 68 6f 6c 20 67 75 65 73 73 20 69 6e 73 70 69 72
0x0075d305 65 20 61 6c 6c 6f 77 20 63 75 70 62 6f 61 72 64
0x0075d315 20 61 6c 61 72 6d 20 6d 6f 75 73 65 20 66 72 65
0x0075d325 73 68 20 63 6f 70 79 20 62 72 69 6e 67 00 00 00

```

FIGURE 15. Firefox extracted volatility hit for password and mnemonic code.

Moreover, there are some cache files left on the file system, but no trace from the web wallets. During the analysis of the RAM, the URL of each web wallet can be found either in ASCII or in UTF-8. Figure 14 shows parts of the located browser history URLs. Figure 15 shows one of the password hits in the RAM and the password is located close to the mnemonic code words, here from the web wallet *greenaddress.it*. In real life, we generally do not know the password, but we can search for the mnemonic code words. In other words, if the mnemonic code words can be found, then there is a good chance to locate the used password. Besides, searching using regular expressions for an email or the wallet ID can potentially return a number of hits, as evidenced by our work in this paper. However, it is challenging to filter the hit list for false positives.

```

Rule: urls
Owner: Process svchost.exe Pid 1156
0x03610f54 62 00 6c 00 6f 00 63 00 6b 00 63 00 68 00 61 00
0x03610f64 69 00 6e 00 2e 00 69 00 6e 00 66 00 6f 00 00 00
0x03610f74 00 00 00 00 46 6a 78 06 79 a7 00 08 00 00 00
Rule: urls
Owner: Process svchost.exe Pid 1156
0x00391802 63 00 6f 00 69 00 6e 00 2e 00 73 00 70 00 61 00
0x00391812 63 00 65 00 00 00 00 00 00 00 00 00 04 b8
0x00391822 d5 14 3d 44 00 14 00 00 00 00 00 00 01 00

```

FIGURE 16. Volatility hits for chrome incognito mode history.

b: CHROME INCOGNITO MODE

The web wallet *greenaddress.it* does not work in Chrome Incognito mode and will not be considered for this test. Similar to Mozilla Firefox, in Google Chrome no trace is stored in the browser history, the browser cache or cookies. But the bookmarks are kept even in Incognito mode. During the analysis of the RAM, each URL of the different web wallets can be found. Figure 16 shows parts of the found browser history URLs. The search with known values like password, email, wallet ID, PIN or also with patterns gives only reasonable hits with the password. Figure 17 shows the found password, which is located close to parts of the mnemonic code from the web wallet *coin.space*. **Internet Explorer InPrivate Browsing**

```

Owner: Process vmtocled.exe Pid 2860
0x02ddc47d 7a 61 65 4a 75 39 69 65 74 61 65 00 00 00 00
0x02ddc48d 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x02ddc49d 00 00 00 00 00 00 00 00 00 01 00 61 63 74 75
0x02ddc4ad 73 20 74 72 69 70 20 73 6c 65 6e 64 65 72 20 65
0x02ddc4bd 6e 68 61 6e 63 65 00 00 00 00 00 00 00 00

```

FIGURE 17. Chrome extracted volatility hit for password and mnemonic code.

```

IECacheView: C:\Users\IEUser\AppData\Local\Microsoft\Windows\WebCache
File Edit View Options Help
File Name Content Type URL
5dc7fe7495[3].js text/javascript cha... https://bam.nr-data.net/1/5dc7fe7495?a=996263...
appstore-3a3e91... image/png https://xapo.com/assets/images/appstore-3a3e9...
client-test[1].json application/json https://nexus-websocket-b.intercom.io/client-test
client-test[2].json application/json https://nexus-websocket-a.intercom.io/client-test
F30BPR43.HTM ?CodeDownloadErrorLogName={00000000-0000-...

```

FIGURE 18. IE 11 inPrivate browsing cache artifacts.

```

Rule: urls
Owner: Process lsass.exe Pid 524
0x01214704 63 6f 69 6e 61 70 75 6c 74 2e 63 6f 6d 82 0d 63
0x01214714 6f 69 6e 61 70 75 6c 74 2e 63 6f 6d 30 0d 06 09
0x01214724 2a 86 48 86 f7 0d 01 0b 05 00 03 82 01 01 00
Rule: urls
Owner: Process svchost.exe Pid 1156
0x03300f8c 62 00 69 00 74 00 67 00 6f 00 2e 00 63 00 6f 00
0x03300f9c 6d 00 00 00 45 6a 78 05 77 a7 00 0e 00 00 00
0x03300fac c4 0f 30 03 00 00 00 03 00 00 28 65 3c 00

```

FIGURE 19. Volatility hits for IE 11 inprivate browsing history.

```

Rule: advanced
Owner: Process vmtocled.exe Pid 2860
0x02d88485 62 69 74 63 6f 69 6e 74 68 65 73 69 73 40 67 6d
0x02d88495 61 69 6c 2e 63 6f 6d 00 00 00 00 00 00 00 00
0x02d884a5 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x02d884b5 00 00 00 00 00 01 00 72 65 62 65 6c 20 70 6f
0x02d884c5 77 65 72 20 68 65 69 67 68 74 20 70 61 6e 64 61
0x02d884d5 20 6d 6f 72 65 20 74 6f 6f 74 68 20 61 6c 63 6f
0x02d884e5 68 6f 6c 20 67 75 65 73 73 20 69 6e 73 70 69 72
0x02d884f5 65 20 61 6c 6c 6f 77 20 63 75 70 62 6f 61 72 64
0x02d88505 20 61 6c 61 72 6d 20 6d 6f 75 73 65 20 66 72 65
0x02d88515 73 68 20 63 6f 70 79 20 62 72 69 6e 67 00 00 00
0x02d88525 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

```

FIGURE 20. IE 11 extracted volatility hit for email and mnemonic code.

Similar to Mozilla Firefox, the web wallet *coin.space* does not work in InPrivate browsing mode and will not be considered for this test. Despite Internet Explorer InPrivate browsing the browser keeps some traces in the browser cache. Figure 18 shows one of the artifacts for the web wallet *xapo.com*, but after restarting the system this cache is cleared. There is no artifact stored in history or cookies, but the favorites are kept. Moreover, Microsoft IE keeps some traces of the browsed URLs on the file system. At the path “C:\Users\IEUser\AppData\Local\Microsoft\Windows\WebCache” several logs which include traces of the web wallets are stored. The different logs include artifacts of every URL, but no saved passwords, PINs or mnemonic codes. During the analysis of the RAM, each URL of the different web wallets can be found too. Figure 19 shows parts of the found browser history URLs. Only one concrete hit for an email which was used to login and some words of the mnemonic code can be found – see Figure 20. Other known values like password, wallet ID, PIN or patterns give no reasonable hit.

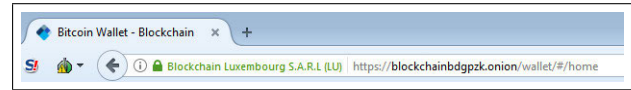


FIGURE 21. TOR URL for the web wallet “blockchain.info”.



FIGURE 22. TOR bookmark Xapo.com.

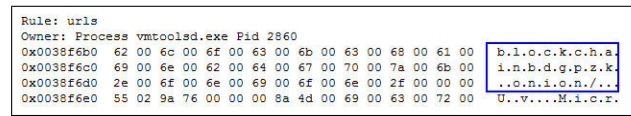


FIGURE 23. Volatility hit for TOR browsing history.

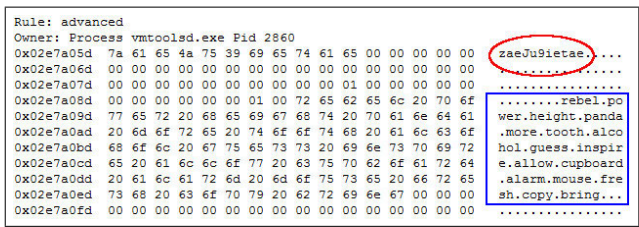


FIGURE 24. TOR extracted volatility hit for password and mnemonic code.

3) TOR BROWSER

The TOR browser also uses SQLite format. However, *bitgo.com*, *coin.space* and *coinbase.com* do not work in the TOR browser, even when cookies are explicitly allowed in the settings. In addition, we need to confirm that we are not a robot via a separate email, where the IP addresses from the client email and the browser are compared. But even if we are successful, the page does not load. The wallet *blockchain.info* forwards to a different URL within the TOR network, which can be seen in Figure 21. The TOR browser deletes the history entries, cookies and the cache. Only bookmarks remain in the table “moz_bookmarks”. Figure 22 shows an extract of the stored bookmark from the web wallet *xapo.com*. During the analysis of the RAM, only the URL of the web wallet *blockchain.info* can be found. Figure 23 shows the found browser history URL. Similar to Mozilla Firefox in private browsing mode, the search using the password gives good hits. Figure 24 shows one of the password hits in the RAM and the password is located close to the mnemonic code words. The search with regular expressions for an email or the wallet ID gives lot of hits, but there is no possibility to filter the hit list for false positives. Other known values like wallet ID, email and PIN give no reasonable hit.

4) BITCOIN WEB WALLETS WITH WINDOWS 10: A COMPARATIVE SUMMARY

a: MOZILLA FIREFOX NORMAL MODE

The artifacts of Mozilla Firefox running on Windows 10 are almost the same as in Windows 7. The only difference is that the tool WebBrowserPassView has no hits in Windows 10. Compared to Windows 7 (Figure 6), this is a key difference.

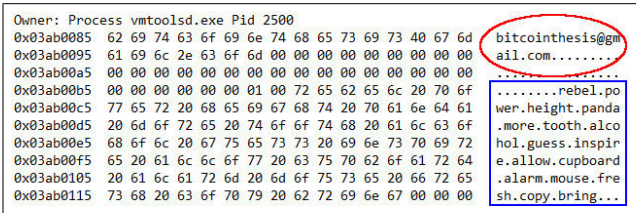


FIGURE 25. Chrome extract volatility hit for email and mnemonic code, Windows 10.

b: MOZILLA FIREFOX PRIVATE BROWSING MODE

Running Mozilla Firefox in private browsing mode on Windows 10 results in no artifact for the history, the cache or the cookies. Similar to Firefox private browsing on Windows 7, browsed URLs, passwords and email addresses are located in the RAM.

c: GOOGLE CHROME NORMAL MODE

There is no difference between Windows 10 and Windows 7 artifacts.

d: GOOGLE CHROME INCOGNITO MODE

Google Chrome Incognito mode has two differences to the use on Windows 7. The web wallet *greenaddress.it* works on Windows 10. Furthermore, the search with known values like password, email, wallet ID, PIN and using patterns gives reasonable hits with the email, not only with the password. This is different on Windows 7. Figure 17 shows the found email, which is located close to parts of the mnemonic code.

e: MICROSOFT INTERNET EXPLORER AND MICROSOFT EDGE NORMAL MODE

On Windows 10, Microsoft Edge is installed by default, in addition to IE. The results of IE in Windows 10 are almost the same as the results in Windows 7. Furthermore, the results of IE and Edge are similar, with the exception of the favorites. Figure 26 shows the favorites from Microsoft Edge, recovered using ESEDatabaseView. The favorites of Internet Explorer can still be found at “C:\Users\Favorites”.

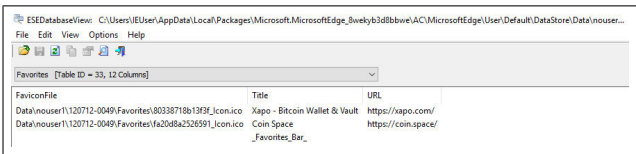


FIGURE 26. Extract NirSoft ESEDatabaseView for microsoft edge.

f: MICROSOFT INTERNET EXPLORER AND MICROSOFT EDGE INPRIVATE BROWSING

With Microsoft Internet Explorer there are no differences between Windows 10 and Windows 7 artifacts. However, Microsoft Edge writes more data to the RAM, also in InPrivate browsing mode. Additionally to found emails, the password is found several times and there is also one hit for the wallet ID which is shown in Figure 27.

```

Owner: Process vmtoolsd.exe Pid 4128
0x04421a09 30 37 36 62 61 36 31 61 2d 35 38 34 37 2d 34 33 076ba61a-5847-43
0x04421a19 31 66 2d 38 66 62 39 2d 66 39 33 65 66 64 38 33 1f-8fb9-f93efd83
0x04421a29 38 66 33 30 00 00 00 00 00 00 00 00 00 00 00 8f30.....
0x04421a39 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

```

FIGURE 27. Extract Windows 10 edge wallet ID.

```

Owner: Process vmtoolsd.exe Pid 4412
0x04220231 30 37 36 62 61 36 31 61 2d 35 38 34 37 2d 34 33 076ba61a-5847-43
0x04220241 31 66 2d 38 66 62 39 2d 66 39 33 65 66 64 38 33 1f-8fb9-f93efd83
0x04220251 38 66 33 30 0d 0a 50 77 3a 20 7a 61 65 4a 75 39 8f30..Pw: zaeJu9
0x04220261 69 65 74 61 65 0d 0a 0d 0a 0d 0a 0d 0a 62 69 74 ietae.....bit
0x04220271 67 6f 2e 63 6f 6d 0d 0a 62 69 74 63 6f 69 6e 74 go.com..bitcoin
0x04220281 68 65 73 69 73 40 67 6d 61 69 6c 2e 63 6f 6d 0d hesis@gmail.com.
0x04220291 0a 50 77 3a 20 7a 61 65 4a 75 39 69 65 74 61 65 .Pw: zaeJu9ietae
0x042202a1 0d 0a 47 6f 6f 67 6c 65 20 41 75 74 68 65 6e 74 ..Google.Authent
0x042202b1 69 63 61 74 6f 72 0d 0a 0d 0a 4f 4c 4a 3a 43 6f icator....OLD:Co
0x042202c1 64 65 3a 20 4d 49 41 51 46 5a 57 59 37 36 5a 42 de: MIAQFZNY76ZB
0x042202d1 41 49 52 5a 4f 49 43 49 53 4a 4a 4e 46 56 56 42 AIR20ICISJDNFVVB
0x042202e1 55 46 33 54 44 46 56 4e 32 5a 51 59 34 48 4c 54 UF3TDFVNI2QY4HLT
0x042202f1 41 33 5a 54 4c 4b 58 51 0d 0a 50 77 3a 20 7a 61 A3ZTLKXQ..Pw: za
0x04220301 65 4a 75 39 69 65 74 61 65 0d 0a 47 6f 6f 67 6c eJu9ietae..Googl
0x04220311 65 20 41 75 74 68 65 6e 74 69 63 61 74 6f 72 3a e.Authenticator:
0x04220321 20 0d 0a 0d 0a 68 74 74 70 73 3a 2f 2f 63 6f 69 .....https://coi

```

FIGURE 28. Extract Windows 10 TOR RAM Hits.

TABLE 6. Overview browser artifacts in normal mode, Windows 7.

Normal Mode	Firefox	Chrome	IE
History	Yes	Yes	Yes
Cookies	Yes	Yes	Yes
Cache	Yes	Yes	Yes
Bookmarks /Favorites	Yes	Yes	Yes
Login Data	Wallet IDs, Email, Mnemonics and Passwords	Wallet IDs, Email, Mnemonics and Passwords	Wallet IDs, Email, Mnemonics and Passwords

TABLE 7. Overview browser artifacts in private mode, Windows 7.

Incognito, InPrivate, Private Mode	Firefox	Chrome	IE	TOR
History	No	No	No	No
Cookies	No	No	No	No
Cache	Restricted	No	Restricted	No
Bookmarks /Favorites	Yes	Yes	Yes	No
Login Data	No	No	No	No
RAM	URLs, Passwords, Mnemonics, Wallet IDs	URLs, Passwords, Mnemonics	URLs, Email, Mnemonics	URLs, Passwords, Mnemonics, Wallet IDs

g: TOR BROWSER

“Normal” artifacts like history, cookies, cache and bookmarks are the same as on Windows 7. However, the analysis of the RAM results in more hits than on Windows 7. All tested web wallets has residual artifacts in the RAM. Figure 28 shows an example of the blockchain.info wallet ID and associated password. Furthermore, for the URL bitgo.com the email address, the associated password and a hint that Google Authenticator was used can be found.

C. FILE SIGNATURES AND KEYWORDS

The list of Bitcoin file signatures garrykessler: sig [13] can also be used to locate bitcoins on a system. Besides, a list of keywords which can also be used for searching and finding traces of Bitcoins in nearly every category of artifacts.

TABLE 8. Differences browser artifacts from Windows 7 to Windows 10.

Browser and Mode	Differences to Windows 7
Firefox Normal Mode	No hits with login data
Firefox Private Browsing	No difference
Chrome Normal Mode	No difference
Chrome Incognito Mode	Additional search with email gives reasonable hits too
IE and Edge Normal mode	No difference
IE and Edge InPrivate	Edge writes more data to the RAM
TOR	Writes a lot of more data to the RAM

TABLE 9. List of Bitcoin keywords.

blockchain	bitgo	coin.space	greenaddress	coinapult	coinbase
xapo	bitcoin	armory	electrum	wallet	keychain
signa	miner	mining	mixer	mixing	laundries
tumblers	easywallet	asmoney	transaction	encrypt	satoshi
testnet	mainnet	altcoin	btcdirect	base58	btcbuckets
freshbtc	hotwallet	vertcoin	witcoin	-	-

Moreover, these lists are “living” lists; thus, they will not be finalized and continue to grow or shrink if a keyword gives poor or too many hits. Additionally, for this list a Webcrawler can be used to extend this list, for example from the page index of the Bitcoin Wiki. The keyword search should not be case sensitive. Table 9 contains a sample list with keywords to search for bitcoin artifacts.

VII. CONCLUSION

Findings from this paper demonstrated that artifacts of forensic interest can be located on systems that had cryptocurrencies installed, for example using an automated tool to search for Bitcoin keys and addresses (e.g. our research identified eight different keys and addresses). The use of prefetch files to identify either the use of a browser or a client to access Bitcoin account is one indicator that can be useful in guiding an investigation. Evidence of uninstallation of Bitcoin-clients can also be used to guide the forensic investigation. In our research, we found that the available list of file signatures relating to Bitcoins is limited (i.e. only nine are known). This can be used as an additional lead, but these files can also be found on the system through other search methods (e.g. keywords or bitcoin-clients). Furthermore, the keyword search can be useful to help us obtain an overview of the system and also older artifacts can be found, because there is no time or storage limit.

We also posit the importance of acquiring and analysing the RAM and the related files pagefile.sys and hiberfil.sys. In addition, for browser artifacts, login information may also be potentially found. In normal mode, all browsers leave a large number of artifacts on the system, and in some cases even the information for the login (e.g. user name and password). In the different private modes, including the TOR browser, fewer number of artifacts are kept on the system (except for bookmarks or favorites). One way of locating the traces is to perform searches in the RAM, the pagefile.sys or the hiberfil.sys. Except when using the TOR

browser on Windows 10, no complete login data (including the often used additional PIN) can be reconstructed. A potential lead is to search for mnemonic code words, which are sometimes located in the RAM after the password. The problem with these kinds of artifacts is, that they often require a two way authentication, e.g. Google Authenticator.

While we believe that our proposed approach can be extended to investigating a broad range of cryptocurrency wallets and systems that have such wallet applications installed, this needs to be experimentally confirmed. Hence, one future research agenda is to investigate a broader range of wallet and browser applications and systems. There are additional data, such as mnemonic code words, the wallet ID or an email address, used either by Bitcoin-clients or web wallets for logging in. To search for wallet IDs or email addresses is not effective, due to the large number of false positives to be filtered manually. As noted in the study of [2], locating mnemonic code words can be used to facilitate the offline brute-forcing of a Bitcoin wallet. Hence, using artificial intelligence (AI) techniques, particularly eXplainable AI (XAI), to facilitate evidence search is another potential future research agenda.

REFERENCES

- [1] A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder, *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*. Princeton, NJ, USA: Princeton Univ. Press, 2016.
- [2] T. Volety, S. Saini, T. McGhin, C. Z. Liu, and K.-K. R. Choo, "Cracking bitcoin wallets: I want what you have in the wallets," *Future Gener. Comput. Syst.*, vol. 91, pp. 136–143, Feb. 2019.
- [3] L. Van Der Horst, K.-K. R. Choo, and N.-A. Le-Khac, "Process memory investigation of the bitcoin clients electrum and bitcoin core," *IEEE Access*, vol. 5, pp. 22385–22398, 2017.
- [4] (2013). *Magnet IEF*. [Online]. Available: <https://www.magnetforensics.com/magnetief/>
- [5] D. Neilson, S. Hara, and I. Mitchell, "Bitcoin forensics: A tutorial," in *Proc. 11th Int. Conf. Global Secur., Saf. Sustainability (ICGS)*, London, U.K., Jan. 2017, pp. 12–26.
- [6] C. W. Cohen. (2015). *Forensics and Bitcoin*. [Online]. Available: <https://articles.forensicrofocus.com/2015/01/16/forensics-bitcoin/>
- [7] A. Montanez, "Investigation of cryptocurrency wallets on iOS and Android mobile devices for potential forensic artifacts," Dept. Forensic Sci., Marshall Univ., Huntington, WV, USA, Tech. Rep., 2014.
- [8] H. Trevor, B. Frank, and B. Ibrahim, "If I had a million cryptos: Cryptowallet application analysis and a trojan proof-of-concept," in *Digital Forensics and Cyber Crime*. New Orleans, LA, USA, Sep. 2018.
- [9] M. Doran, "A forensic look at bitcoin cryptocurrency," Ph.D. dissertation, SANS Inst., Bethesda, MD, USA, 2015.
- [10] S. Ali, A. ElAshmawy, and A. F. Shosha, "Memory forensics methodology for investigating cryptocurrency protocols," in *Proc. Int. Conf. Secur. Manage.*, Las Vegas, NE, USA, Aug. 2018, pp. 153–159.
- [11] A. M. Antonopoulos, *Mastering Bitcoin*. Newton, MA, USA: O'Reilly Media, 2014, ch. 4.
- [12] G. van Rossum, B. Warsaw, and N. Coghlan. (2001). *Style Guide for Python Code*. [Online]. Available: <https://www.python.org/dev/peps/pep-0008/#naming-conventions>
- [13] G. C. Kessler. (2017). *File Signatures Table*. [Online]. Available: http://www.garykessler.net/library/file_sigs.html
- [14] C. Warren, E. El-Sheikh, and N.-A. Le-Khac, "Privacy preserving Internet browsers: Forensic analysis of Browzar," in *Computer and Network Security Essentials*, K. Daimi, Eds. New York, NY, USA: Springer, 2017.

...