BitIodine: Extracting Intelligence from the Bitcoin Network

Michele Spagnuolo, Federico Maggi, and Stefano Zanero

Politecnico di Milano, Italy
michele.spagnuolo@mail.polimi.it, federico.maggi@polimi.it, stefano.zanero@polimi.it

Abstract. Bitcoin, the famous peer-to-peer, decentralized electronic currency system, allows users to benefit from pseudonymity, by generating an arbitrary number of aliases (or addresses) to move funds. However, the complete history of all transactions ever performed, called "blockchain", is public and replicated on each node. The data it contains is difficult to analyze manually, but can yield a high number of relevant information.
In this paper we present a modular framework, BitIodine, which parses the blockchain, clusters addresses that are likely to belong to a same user or group of users, classifies such users and labels them, and finally visualizes complex information extracted from the Bitcoin network.
BitIodine labels users (semi-)automatically with information on their identity and actions which is automatically scraped from openly available information sources. BitIodine also supports manual investigation by finding paths and reverse paths between addresses or users.
We tested BitIodine on several real-world use cases, identified an address likely to belong to the encrypted Silk Road cold wallet, or investigated the CryptoLocker ransomware and accurately quantified the number of ransoms paid, as well as information about the victims.
We release an early prototype of BitIodine as a library for building more complex Bitcoin forensic analysis tools.

Keywords: Bitcoin, financial forensics, blockchain analysis

1 Introduction

Bitcoin is a decentralized monetary system based on an open-source protocol and a peer-to-peer network of participants that validates and certifies all transactions. It aims to become the digital equivalent of cash, as its transactions do not explicitly identify the payer nor the payee.

Some features of Bitcoin, such as cryptographically guaranteed security of transactions, negligible transaction fees, no set-up costs and no risk of charge-back, along with its surging conversion rates to USD, convinced several businesses to adopt it. At the same time, its apparent anonymity and ease of use attracted also cybercriminals [5], who use it as a way of monetizing botnets and extorting money (e.g., we discuss the CryptoLocker case in §4.3).

The decentralized accounting paradigm typical of Bitcoin requires each node of the network to keep in memory the entire history of every transaction ever

happened, called *blockchain*. Although Bitcoin identities are not explicitly tied to real-world individuals or organizations, all transactions are public and transparent, and since each one is tied to the preceding one(s), anyone can see the flow of Bitcoin from address to address.

Some bitcoin addresses are known and tied to entities such as gambling sites, forum users or marketplaces. By analyzing the blockchain and correlating it with this publicly available meta data, it is possible to find addresses used (e.g., for gambling, mining, or for scams). Addresses can be algorithmically grouped in clusters that correspond with entities that control them (but do not necessarily *own* them) [1,2,5,9]. Collapsing addresses into clusters simplifies the huge transaction graph, creating edges that correspond to aggregate transactions (i.e., money exchanges) between entities or users. From hereinafter we refer to such clusters and entities as *users*. The interesting outcome for investigators is that it is possible to simply retrieve valuable information about an entity from one of its addresses.

In existing approaches, clusters are labeled mostly manually, and the whole process is not automated. In this paper, we propose BitIodine, a collection of modules to automatically parse the blockchain, cluster addresses, classify addresses and users, graph, export and visualize elaborated information from the Bitcoin network. In particular, we devise and implement a classifier module that labels the clusters in an automated or semi-automated way, by using several web scrapers that incrementally update lists of addresses belonging to known identities. We create a feature-oriented database that allows fast queries about any particular address to retrieve balance, number of transactions, amount received, amount sent, and ratio of activity concerning labels (e.g., gambling, mining, exchanges, donations, freebies, malware, FBI, Silk Road), or, in an aggregated form, for clusters. It is possible to query for recently active addresses, and filter results using cross filters in an efficient way.

BitIodine has been tested on several real-world use cases: we describe how we used it to find the transaction that, according to the FBI, was a payment by Dread Pirate Roberts, founder of the Silk Road, to a hitman to have a person killed [11]. We find a connection between Dread Pirate Roberts and an address with a balance exceeding 111,114 BTC[1], likely belonging to the encrypted Silk Road cold wallet. Finally, we investigate the CryptoLocker ransomware, and, starting by an address posted on a forum by a victim, we accurately quantify the ransoms paid (around 1226 BTC as of December 15, 2013), and get information about the victims.

In summary, our contributions are:

- A future-proof and easily extendable framework for building complex applications for forensic analysis of the Bitcoin blockchain: `http://miki.it/downloads/bitiodine.zip`.
- A system that labels clusters/users with little or no supervision.
- We test our system on real-world use cases that include investigations on the Silk Road and on malware such as CryptoLocker.

[1] The common shorthand currency notation for Bitcoin(s)

2 State of the art and motivation

Bitcoin transactions [7] do not explicitly identify payers nor payees, as they are just cryptographically signed messages that "encode" a fund transfer from one public key to another. No PKI is present. The private keys are needed to authorize such transfer.

The decentralized paradigm of Bitcoin requires each node of the network to retain the *blockchain* (i.e., entire transaction history). All transactions are public, transparent, and permanently recorded since the origin. Therefore, a lot of potentially interesting information can be mined from the blockchain. Some addresses are known and tied to entities, such as for instance gambling sites, users of the main Bitcoin-related forum, Bitcoin Talk, or Bitcoin-OTC marketplace. By analyzing the blockchain, it is possible to automatically find out how much an address is used for gambling activities or mining, if it was used for scamming users in the past, if and how it is related to other addresses and entities. The idea of algorithmically associating Bitcoin addresses to entities controlling them is described in [1] and [2]. The first work investigates Bitcoin privacy provisions in a simulated setting where Bitcoin is used for daily payments, and concludes that the current implementation of Bitcoin would enable the recovery of user transaction profiles to a large extent. The second work analyzes the Bitcoin network with data mining and anomaly detection techniques, using simple network features, to monitor the network for identify thefts.

Reid et al. [9] analyzed the anonymity in Bitcoin and advocated the need for proper PKI-like mechanisms. The activity of known users can be observed in detail using passive analysis only, but the authors take into consideration also active analysis, where an interested party can potentially deploy *marked* Bitcoins and collaborate with other users to discover even more information. Mixing services (e.g., Bitcoin Fog) claim to obfuscate the origin of transactions, thus increasing the users' anonymity: their effectiveness is analyzed in [6]. Structural patterns in the topology and dynamics of the Bitcoin transaction graph that have implications for the users' anonymity are shown in [8], whereas [3] collected precious information about the Silk Road before the seizure by the FBI.

The forensic approach proposed in [5] focuses on investigating the use of Bitcoin for criminal or fraudulent purposes at scale. Using a small number of manually labeled transactions, the authors were able to identify major institutions and the interactions between them, and demonstrated that this approach can shed light on the structure of the Bitcoin economy and how it is used.

From the state of the art we conclude that labeling addresses and users automatically will become a necessity as the Bitcoin network grows.

3 System design and implementation

BitIodine is meant to be a modular, expandable and easily deployable framework to build complex applications for forensic analysis of the Bitcoin blockchain.

3.1 Architecture and data flow overview

Fig. 1 describes in a simplified way the building blocks of BitIodine and the interactions between different modules.

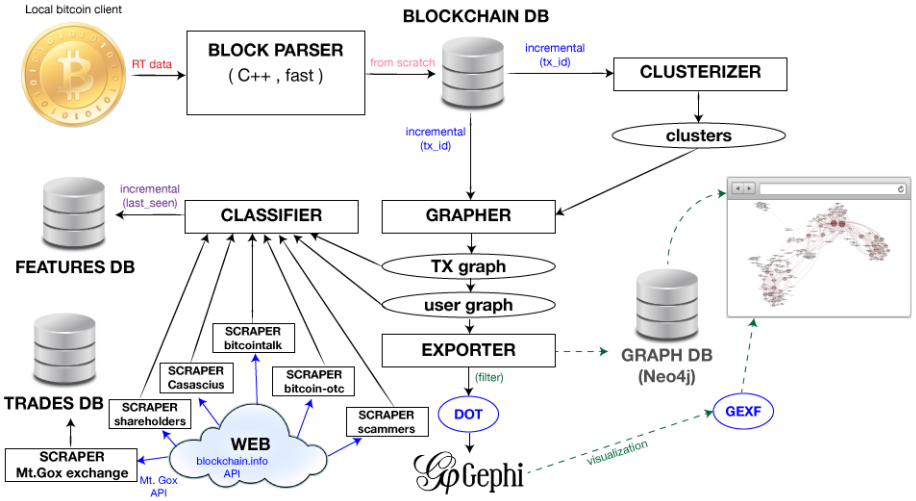

Fig. 1: Building blocks of BitIodine

The **Block Parser** reads blocks and transactions from the local `.bitcoin` folder populated by the official *bitcoind* client and exports the blockchain data to the *blockchain DB*, which uses a custom relational schema that we designed to obtain good performance (see §3.3). This allows for a fast updating of data from the Bitcoin network.

The goal of the **Clusterizer** is to find groups of addresses that belong to the same user. It incrementally reads the blockchain DB and generates-updates clusters of addresses using two heuristics, detailed in §3.2. The first heuristic exploits transactions with multiple inputs, while the second leverages the concept of "change" in transactions (see §3.2). These clusters are stored in *cluster files*.

A set of **Scrapers** crawl the web for Bitcoin addresses to be associated to real users, automatically collecting, generating and updating lists of:

- *usernames* on platforms, namely *Bitcoin Talk* forum and *Bitcoin-OTC* marketplace (from forum signatures and databases)
- *physical coins* created by Casascius (`https://www.casascius.com`) along with their Bitcoin value and status (opened, untouched)
- known *scammers*, by automatically identifying users that have significant negative feedback on the Bitcoin-OTC and Bitcoin Talk trust system.
- *shareholders* in stock exchanges (currently limited to *BitFunder*)

Additional lists can be built with a semi-automatic approach which requires user intervention. In particular, by downloading tagged data from `https://blockchain.info/tags`, the tool helps users build lists of *gambling* addresses, *online wallet* addresses, *mining pool* addresses and addresses which were subject to *seizure* by law enforcement authorities. The user can verify tags and decide to put the most relevant ones in the correct lists. Finally, a scraper uses Mt. Gox trading APIs to get historical data about trades of Bitcoin for US dollars, and saves them in a database called *trades DB*. This module is useful to detect

interesting flows of coins that enter and exit the Bitcoin economy. The interface is easily expandable, and adding scrapers for new services and websites is easy.

The **Grapher** incrementally reads the *blockchain DB* and the *cluster file* to generate, respectively, a *transaction graph* and a *user graph*. In a transaction graph, addresses are nodes and single transactions are edges. The **Grapher** has several applications (e.g., finding successors and predecessors of an address). In a user graph, users (i.e., clusters) are represented as nodes, and the aggregate transactions between them are represented as edges.

The **Classifier** reads the *transaction graph* and the *user graph* generated by the *grapher*, and proceeds to automatically label both single addresses and clusters with specific annotations. Examples of labels are Bitcoin Talk and Bitcoin-OTC usernames, the ratio of transactions coming from direct or pooled mining, to/from gambling sites, exchanges, web wallets, other known BitcoinTalk or Bitcoin-OTC users, freebies and donation addresses. There are also boolean flags, such as *one-time address*, *disposable*, *old*, *new*, *empty*, *scammer*, *miner*, *shareholder*, *FBI*, *Silk Road*, *killer* and *malware*. A complete list can be found in [10]. Classification can take place globally on the whole blockchain, or selectively on a list of specified addresses and clusters of interest. The results are stored in a database and can be updated incrementally.

The **Exporter** allows to export and filter (portions of) the *transaction graph* and the *user graph* in several formats, and support manual analysis by finding *simple paths* (i.e., paths with no repeated nodes) on such graphs. More precisely, it can export transactions that occurred inside a cluster, or that originated from a cluster. It can also find either the shortest, or all the simple paths from an address to another address, from an address to a cluster, from a cluster to an address, or between two clusters. Moreover, it can find all simple paths originating from an address or a cluster (i.e., the subgraph of successors), or to reverse such search, by identifying the subgraph of predecessors of an address or cluster. Subgraphs of successors or predecessors can be useful, for instance, in taint analysis, and can assist manual investigation of mixing services, as we do in §4.1.

3.2 Algorithms and analysis approaches

Let N be the whole Bitcoin network. We denote with n_B, n_U, n_A, respectively, the total number of blocks, users and addresses in the network. We also denote as $B = \{b_1, b_2, \ldots, b_{n_B}\}$ the set of blocks in the network N, and similarly as $U = \{u_1, u_2, \ldots, u_{n_U}\}$ the set of users and as $A = \{a_1, a_2, \ldots, a_{n_A}\}$ the set of addresses. We also denote with $\tau_i(S_i \to R_i)$ a transaction with a unique index i, and $S_i \subseteq A$ and $R_i \subseteq A$ denote the sets of senders' addresses and recipients' addresses, respectively. We define $T = \{\tau_1(S_1 \to R_1), \tau_2(S_2 \to R_2), \ldots, \tau_{n_T}(S_{n_T} \to R_{n_T})\}$ as the set of all n_T transactions which took place. We also define $T|_{b_i} \subset T$ as the subset of all the transactions contained in blocks with index $k \leq i$. Blocks are uniquely identified by indexes starting from 0, for the *genesis block*, sequentially increasing as they are appended to the blockchain.

We also define two functions. *lastblock*: $T \mapsto B$, a function that maps the set of transactions to the set of blocks, such that $lastblock(\tau_i) = b_i$ if and only if b_i is the last block relayed by the network N as the transaction τ_i is broadcast.

owns: $A \mapsto U$, a function that maps the set of addresses to the set of users, such that $owns(a_i) = u_k$ if and only if u_k owns the private key of a_i.

First heuristic: Multi-input transactions grouping The first heuristic exploits multi-input transactions. Multi-input transactions occur when a user u wishes to perform a payment, and the payment amount exceeds the value of each of the available Bitcoin in u's wallet. In order to avoid performing multiple transactions to complete the payment, enduring losses in terms of transaction fees, Bitcoin clients choose a set of Bitcoin from u's wallet such that their aggregate value matches the payment and perform the payment through multi-input transactions. This means that whenever a transaction has multiple input addresses, we can safely assume that those addresses belong to the same wallet, thus to the same user.

More formally, let $\tau_i(S_i \to R_i) \in T$ be a transaction, and $S_i = \{a_1, a_2, \ldots, a_{n_{S_i}}\}$ the set of input addresses. Let also $|S_i| = n_{S_i}$ be the cardinality of the set. If $n_{S_i} > 1$, then all input addresses belong to the same (previously known or unknown) user: $owns(a_i) \triangleq u_k \quad \forall i \in S_i$.

Second heuristic: shadow address guessing The second heuristic has to do with *change* in transactions. The Bitcoin protocol forces each transaction to spend, as output, the whole input. This means that the "unspent" output of a transaction must be used as input for a new transaction, which will deliver "change" back to the user. In order to improve anonymity, a *shadow address* is automatically created and used to collect the change that results from any transaction issued by the user. The heuristic tries to predict which one of the output addresses is actually belonging to the same user who initiated the transaction, and it does so in two possible ways: the first one is completely deterministic, the second one exploits a (recently fixed) flaw in the official Bitcoin client.

The completely deterministic and conservative variant works as follows: If there are two output addresses (one payee and one change address, which is true for the vast majority of transactions), and one of the two has never appeared before in the blockchain, while the other has, then we can safely assume that the one that never appeared before is the shadow address generated by the client to collect change back.

More formally, let $\tau_i(S_i \to R_i) \in T$ be a transaction, and $R_i = \{a_1, a_2, \ldots, a_{n_{R_i}}\}$ be the set of output addresses (with $|R_i| = n_{R_i}$ being the cardinality of the set), and let us consider $T|_{lastblock(\tau_i)}$, that is, the set T limited to the last block at the time of transaction τ_i. If $n_{R_i} = 2$, then the output addresses are a_1 and a_2. If $a_1 \notin T|_{lastblock(\tau_i)}$ and $a_2 \in T|_{lastblock(\tau_i)}$, then a_1 is the shadow address, and belongs to the same user u_k who owns the input address(es): $owns(a_1) \triangleq u_k$.

A bug in the `src/wallet.cpp` file of the official Bitcoin client allows to improve upon this heuristic. When the client chooses in which slot to put the shadow address, it passes to `GetRandInt` the number of payees. Thanks to an off-by-one error, in the common case of one payee, `GetRandInt` will always return 0, and the change always ends up in the first output.

For transactions prior the fix was released (Feb 3, 2013), only 6.8% have the shadow address provably in the second slot of two-outputs transactions. Therefore,

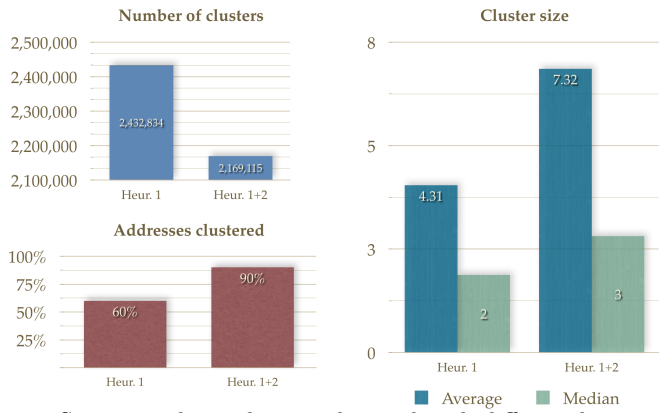

Fig. 2: Statistics about clusters obtained with different heuristics.

for transactions before this date we can relax the heuristic, and consider a first output that was previously unseen in any two-output transaction as a shadow address, regardless of the second one. This allows for a much better coverage, and generates much more compact clusters of users, as shown in Fig. 2.

3.3 Implementation details

Since BitIodine deals with several gigabytes of data, and graphs with millions of nodes and tens of millions of edges, there are significant scalability and performance issues to overcome.

We used Python 3.3.3rc1 for every module, except the `Block Parser`, which is written in C++ for performance reasons. The block parser is a modified version of the *blockparser* tool by *znort987*[2], to which we added several custom callbacks: our modified version is highly efficient in exporting all addresses on the network, in performing taint analysis on an address, and in exporting to SQLite.

We opted for the use of embedded SQLite databases for storing the blockchain and the features database because it is a zero-configuration, server-less, embedded, stable and compact cross-platform solution. We do not need concurrency while writing to database files, so the only possible disadvantage does not affect its use in BitIodine. In designing the custom database schema for BitIodine we had to find a good balance between size and performance, weighing the use of indexes (see §4.4).

The **Clusterizer** is designed to be incremental, and it is also possible to pause the generation of clusters at any time, and resume it from where it stopped.

Internally, graphs are handled by NetworkX, a Python library to create, manipulate, and study the structure, dynamics, and functions of complex networks. NetworkX objects can be serialized and written to a file with ease, and in-memory querying for successors and predecessors of nodes is efficient. Is it also possible to embed an arbitrary number of additional data labels to nodes and edges (e.g., we added transaction hashes).

[2] `http://github.com/znort987/blockparser`

Our **Exporter** supports a multiplicity of output formats, allowing results to be fed into visualization software such as Gephi or exported to a graph database such as Neo4j.

4 Experiments and case studies

The goal of our experiments is to evaluate the correctness (§4.1, 4.2, 4.3) and the performance (§4.4) of BitIodine. Since BitIodine builds novel knowledge, there is no ground truth data available to validate our findings. However, we were able to confirm our findings thanks to contextual information found on the web resources cited in each case study.

4.1 Investigating activity involving Dread Pirate Roberts

On October 1st, 2013, 29-year-old Ross William Ulbricht was arrested on suspicion of being the creator and operator of the infamous "Silk Road" black market, under the alias of "Dread Pirate Roberts" (DPR) [11]. From February 6th, 2011 to July 23rd, 2013, sales through the market totaled 9,519,664 BTC (spread across 1,229,465 transactions), of which 614,305 BTC went directly to the accused as commissions. Prosecutors said they seized approximately 173,600 BTC (at date equivalent to approximately USD 30,000,000), in the largest seizure of the digital currency ever.

The seizure happened in two phases. At first, 29,600 BTC held in a so called *hot wallet* (i.e., an operating pool for the website) were seized. But Ulbricht held the majority of his funds separately in an encrypted "cold wallet". On October 25th, an additional 144,000 BTC were seized (an approximate amount of USD 120,000,000).

The seizure was operated by transferring the seized coins to two addresses controlled by the FBI. These addresses are publicly known[3]. On the other hand, the addresses which formed the cold wallet are not public yet (as of January 28, 2014).

Using BitIodine alone, we are able to find an interesting connection between an address known to belong to DPR and `1933phfhK3ZgFQNLGSDXvqCn32k2buXY8a`, an address with a balance exceeding 111,114 BTC (more than USD 22,000,000), likely belonging to the cold wallet. The investigation is as follows. DPR used to post on the Bitcoin Talk forum as *altoid*: the message at `https://bitcointalk.org/index.php?topic=47811.0` seeks a "*lead developer ... [for a] Bitcoin startup*", and refers to his email address (`rossulbricht@gmail.com`). In a previous post (`https://bitcointalk.org/index.php?topic=6460.msg94424`), he asked help on the PHP Bitcoin API, pasting one of his addresses, `1LDNLreKJ6GawBHPgB5yfVLBERi8g3SbQS`, as a parameter of `sendfrom` method. This can be found out by manual investigation.

By running BitIodine on these data points, we found that Ulbricht's known address belongs to a cluster of 6 addresses, all empty. Thanks to our *path finders* in the **Exporter** module, we automatically found a connection between the leaked address and a very wealthy address, `1933phfhK3ZgFQNLGSDXvqCn32k2buXY8a`, as shown in Fig. 3.

[3] `1F1tAaz5x1HUXrCNLbtMDqcw6o5GNn4xqX, 1FfmbHfnpaZjKFvyi1okTjJJusN455paPH`

```
-> first input transaction of the address on the right
-> only input transaction of the address on the right
-> only significative input transaction of the address on the right
-> address on the left spent all its coins to address on the right exclusively

          1LDNLreKJ6GawBHPgB5yfVLBERi8g3SbQS
->->      1BG9jDV3pA1MsJUnvRyWuA2b7PfGd4MZaw
          5000 BTC 2011-04-30 18:32:55
->->      12h6TzwPNBvDnppbsqpyXwW4oo5UUKaKSa
          2000 BTC 2011-05-07 14:12:51 in a multi-input TX for 9067.32 BTC
->->->    1EG9HJG9aGqzgGujfNQMiNbyqpKnFxafvE
          9067.32 BTC 2011-06-19 23:04:29 in a multi-input TX for 37420.09314115 BTC
->->->    1AHki5AbZYiz4fHkGSTVKN3T1Tv5PwZpnh
          37420.09314115 BTC 2011-06-19 23:29:01 in a multi-input TX for 37421.09314115 BTC
->->      15TEAwEMxVS3BK718HhwgJg7nxwyJ2ib9y
          37421.09314115 BTC 2011-06-22 02:48:45
->->      1933phfhK3ZgFQNLGSDXvqCn32k2buXY8a
          37421.09314115 BTC 2011-07-02 02:42:15 in a multi-input TX for 40954.56541907 BTC
```

Fig. 3: Connection between DPR's address and a 111,114 BTC address

The chain is particularly interesting because every address appears in the blockchain with its first input coming from the previous one in the chain, and often addresses spend all their inputs to addresses on the right exclusively. In our opinion, this is a manual, rudimentary mixer or tumbler, and BitIodine found a meaningful connection between the addresses, leading us to argue (with some grounding) that `1933` was part of the cold wallet of the Silk Road.

Although in this scenario there is some manual investigation, it would have been difficult to find significant links manually, given the millions of nodes involved.

4.2 Payment to a killer?

In March 2013, the Silk Road vendor *FriendlyChemist* supposedly attempted to blackmail DPR via Silk Road's private message system, providing proof that he had names and addresses of thousands of vendors. He demanded USD 500,000 for his silence. DPR asked another user, *redandwhite*, to "execute" FriendlyChemist, supplying him/her his full name and address. On March 31st, 2013, after having agreed on terms, DPR sent redandwhite 1,670 BTC to have FriendlyChemist killed.

Using BitIodine, we easily identify the transaction[4] to the alleged hitman, by querying the blockchain DB for transactions of 1,670 BTC on that day. The killer's address is `1MwvS1idEevZ5gd428TjL3hB2kHaBH9WTL`. This 1,670 BTC transaction is the first input it receives. On April 8, 2013 it receives another 3,000 BTC, and on April 12, 2013 another 2,555 BTC. Investigators could not find any record of somebody in that region being killed around that date or matching that description. This possibly implies that DPR was scammed, and that he was not the only one.

In this use case, BitIodine helps the investigation by allowing to filter transactions by amount and date in an efficient way. Remarkably, having no addresses nor transaction hashes, it would have been hard to spot the transaction manually.

4.3 Ransomware investigation with BitIodine

CryptoLocker [4] is a recent ransomware that encrypts the victim's personal files with strong encryption. The criminals retain the only copy of the decryption key

[4] `4a0a5b6036c0da84c3eb9c2a884b6ad72416d1758470e19fb1d2fa2a145b5601`

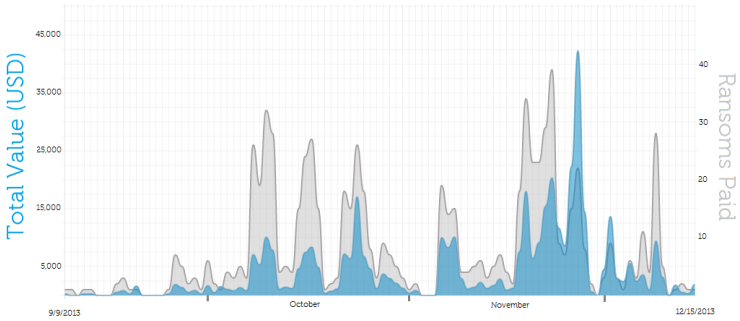

Fig. 4: CryptoLocker infection rate (computed in [4]) plotted vs. ransoms paid in Bitcoin, computed with BitIodine

on their server and ask for a ransom to be paid with MoneyPak or Bitcoin within 72 hours in order to de-crypt the files.

We used BitIodine to detect the CryptoLocker cluster(s), belonging to the malware authors, and compute some statistics about ransoms paid by the victims. By searching on Google for extracts of the text in the request by the malware and by reading a Reddit thread where victims and researchers post addresses[5], we collected several addresses that were known to belong to CryptoLocker. The **Classifer** confirmed that they belonged to several clusters, which comprised a total of 2118 addresses. We identified 771 ransoms, for a total of 1226 BTC (approximately USD 1,100,000 on December 15, 2013). Some addresses received a single payment, others were reused for several ones. Tables listing the detailed data are in [10].

Dell SecureWorks Counter Threat Unit Research Team have been monitoring the CryptoLocker botnet since Sep 18, 2013 and analyzed various data sources, including DNS requests, sinkhole data, and client telemetry, publishing a report [4] overlaying daily infection rates to the ransoms in Bitcoin detected by BitIodine (§4). Spikes coinciding with Cutwail spam campaigns that resulted in increased CryptoLocker infections are indicated in the overlay, including the period of high activity from October through mid-November. Likewise, periodic lulls in activity have occurred frequently, including a span from late November through mid-December.

Finally, it is interesting to analyze the cluster related to the very first ransom paid[6], on Sep 13, four days before the others, because it could be some sort of "test" of the payment mechanism by the malware authors. BitIodine was not able to associate that cluster to a known identity due to a lack of useful data for that particular cluster. Manual analysis confirmed that no known nickname is linked to addresses belonging to that cluster. Therefore, this is not a limitation of our approach: the cluster might get labelled in the future as new transactions are broadcast.

[5] `http://www.reddit.com/r/Bitcoin/comments/1o53hl/disturbing_bitcoin_virus_encrypts_instead_of/`

[6] `http://tinyurl.com/cl-first-ransom`

4.4 Performance evaluation

The generation of the database takes approximately 30 minutes on a Quadruple Extra Large High-Memory AWS EC2 instance (26 ECU, 68.4 GB of RAM), and its size is around 15GB.

The **Clusterizer** generates 4,077,114 clusters, grouping together 18,153,279 addresses, and takes approximately 45 minutes to process the whole blockchain using the same machine.

Scalability issues may arise as the blockchain grows, in particular for operations involving the transaction graph, which has to be loaded in memory. A solution would be to move the graphs to a graph database such as Neo4j, at the expense of slower queries (because of slower disk I/O with respect to memory) and a space occupation on disk almost five times higher. In our tests, a transaction graph updated to November 1, 2013 is 7 GB in NetworkX format and more than 30 GB with a Neo4j database. While Neo4j, thanks to the Cypher Query Language, allows complex queries that fully exploit graph structures, we opted for a simpler and leaner in memory solution at this stage.

5 Limitations and future work

The main limitation is that the first heuristic presented in §3.2 works under the assumption that owners do not share private keys. This does not always hold: for example, some web wallets have pools that would be mistakenly grouped as a single user. This is why we defined the *owns* relation as $owns(a_i) = u_k$ if and only if u_k owns the private key of a_i.

Moreover, the current implementation of the **Classifier** module needs to load the transaction graph and the clusters in memory, making classification a memory-intensive task. Also, BitIodine keeps data in two different fashions: in a relational database (the blockchain and features database) and in a graph (transaction and user graphs). This can be seen as redundant. In a future release, a single, efficient graph solution could replace the relational blockchain DB. In general, we see an on-disk graph database such as Neo4j needed if BitIodine is used in production, even with the drawbacks detailed in §3.3.

Furthermore, currently we label users in a (semi-)automated way by scraping information on known addresses from the web. In future extensions of this work, we envision to mine behavioral patterns of users on the network with unsupervised clustering or classification techniques.

6 Conclusions

BitIodine is a modular framework that parses the Bitcoin blockchain, clusters addresses likely to belong to a same entity, classifies such entities and labels them, and visualizes complex information extracted from the Bitcoin network. Using BitIodine it is possible to label users and addresses automatically or semi-automatically thanks to scrapers that crawl the web and query exchanges for information, thus allowing to attach identities to users and to trace money entering and exiting the Bitcoin economy. BitIodine also supports manual

investigation by finding paths and reverse paths between two addresses or a user and an address.

We tested BitIodine on several real-world use cases. We discovered a connection between the founder of Silk Road and an address likely belonging to the encrypted Silk Road cold wallet. We found the transaction that, according to the FBI, was a payment by the founder of Silk Road to a hitman. Finally, we investigated the CryptoLocker ransomware, and starting by an address posted by a victim, we accurately quantified the number of ransoms paid, and got information about the victims, with very limited manual analysis. Even at this early stage of development, we were able to support investigation of malware activity.

We released BitIodine to allow the community of researchers to enhance it. Our hope is that it can become the framework for building more complex Bitcoin forensic analysis tools. For example, an engineer at Banca d'Italia (Italy's central bank) is currently developing, using BitIodine as a base, *VIREXBC* (*Visual Interactive REaltime eXplorer*), a realtime visualization software of the Bitcoin blockchain for interactively presenting complex imagery and infographics generated on-the-fly.

References

1. Androulaki, E., Karame, G., Roeschlin, M., Scherer, T., Capkun, S.: Evaluating user privacy in bitcoin. In Sadeghi, A.R., ed.: Financial Cryptography and Data Security. Volume 7859 of LNCS., Springer (2013) 34–51
2. Brugere, I.: Anomaly detection in the bitcoin transaction network. Technical report, ESP-IGERT (2012)
3. Christin, N.: Traveling the silk road: a measurement analysis of a large anonymous online marketplace. In: Proc. of the 22nd int.l conf. on World Wide Web. WWW '13 (2013) 213–224
4. Jarvis, K.: CryptoLocker Ransomware. Available online (2013)
5. Meiklejohn, S., Pomarole, M., Jordan, G., Levchenko, K., McCoy, D., Voelker, G.M., Savage, S.: A fistful of bitcoins: characterizing payments among men with no names. In: Proceedings of the 2013 Internet Measurement Conference, ACM (2013) 127–140
6. Möser, M.: Anonymity of bitcoin transactions: An analysis of mixing services. In: Proceedings of Münster Bitcoin Conference. (2013)
7. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. available online (2008)
8. Ober, M., Katzenbeisser, S., Hamacher, K.: Structure and anonymity of the bitcoin transaction graph. Future Internet **5**(2) (2013) 237–250
9. Reid, F., Harrigan, M.: An analysis of anonymity in the bitcoin system. In: Security and Privacy in Social Networks, Springer (2013) 197–223
10. Spagnuolo, M.: Bitiodine: Extracting intelligence from the bitcoin network. Master's thesis, Politecnico di Milano (December 2013)
11. U.S. District Court, Southern District of New York: Alleged silk road founder ross ulbricht criminal complaint. available online (2013)