Threat Analysis of Poisoning Attack Against Ethereum Blockchain

Teppei Sato[1(✉)], Mitsuyoshi Imamura[1], and Kazumasa Omote[1,2]

[1] University of Tsukuba, 1-1-1, Tennodai, Tsukuba 305-8573, Japan
s1820583@s.tsukuba.ac.jp, ic140tg528@gmail.com, omote@risk.tsukuba.ac.jp
[2] National Institute of Information and Communications Technology, 4-2-1, Nukui-Kitamachi, Koganei, Tokyo 184-8795, Japan

Abstract. In recent years, blockchain technology has witnessed remarkable developments in its application to crypto assets (cryptocurrency) considering not only function storing values but also extension of the smart contract and anonymity improvement. Ethereum is a blockchain that features the smart contract and there is a data space, where programs can be freely stored, on the blockchain. However, pollution of such data space can jeopardize the existence of Ethereum.

In this study, we analyze the fact that the malicious files that are stored in the data space of Ethereum and discuss "blockchain poisoning attacks" that significantly contaminate the blockchains by embedding malicious data at a relatively lower cost. We try to tackle Ethereum-specific risks which are not mentioned in previous study. In addition, we empirically examine the possibility of a poisoning attack on a private blockchain network.

Keywords: Blockchain · Crypto assets · Security · Poisoning attack

1 Introduction

Blockchain is a distributed ledger technology that is a part of Bitcoin [16]. It was introduced in 2008. While the technology is well known as the base technology of crypto assets (cryptocurrency), it also acts as not only the function to store values, extending into industrial fields. The technology has some advantages. It provides a certain degree of anonymity, ensures that the network is not shut down, and manipulation-resistance to the data on it. Some researchers proposed its applications in various fields such as IoT security [7], PKI [8], and management of medical data [4]. However, the blockchain technology faces certain security problems and challenges, including typical attacks [9,13] such as the majority attack (51% Attacks), double-spending and cryptojacking, and new types of attacks such as the "blockchain poisoning attack", which can be a critical treat to the blockchain system (Fig. 1).

Blockchains can be attacked by embedding malicious or illegal files in the flexible space of blockchain [15]. We define this type of attack as a blockchain

© IFIP International Federation for Information Processing 2020
Published by Springer Nature Switzerland AG 2020
M. Laurent and T. Giannetsos (Eds.): WISTP 2019, LNCS 12024, pp. 139–154, 2020.
https://doi.org/10.1007/978-3-030-41702-4_9

poisoning attack. Such poisoning attacks are considered to be more malicious than conventional poisoning attacks, such as DNS cache poisoning, against public databases because repairing a poisoned blockchain without hard fork is not feasible owing to the feature of blockchains where the transactions inside a blockchain cannot be modified or cancelled by anyone. In addition, because the data contained in the blockchain are synchronized by each node, attackers can force the nodes to download any malicious files by embedding them into the blockchain.

Smart contract, which was first proposed by Szabo [20], is a computer protocol designed to digitally facilitate, verify, or enforce the negotiation of a transaction without any trusted third parties. Ethereum is well known as a implementation of smart contract which is turing-complete.

Unlike Bitcoin blockchain, Ethereum has a legitimate and flexible space that contains the bytecodes of smart contracts. Anyone can officially embed any data into Ethereum blockchain. Unfortunately, this indicates that this feature also provides flexibility to attackers. Hence, the poisoning attacks against Ethereum blockchain (and blockchains that have a flexible space like Ethereum) are easier.

In this study, we analyze the Ethereum blockchain to examine blockchain poisoning, and further verify the ease of blockchain poisoning attack using our experimental blockchain environment.

The contributions of this paper are following:

- We analyzed the Ethereum blockchain to examine the actual situation of blockchain poisoning (until December 31, 2018 UTC), and found 154 files including some malicious files.
- Using experimental blockchain environments, we demonstrated that blockchain poisoning can be easily done through web browsers and one-liner shell command. This risk is specific to Ethereum, not discussed in previous study [15].
- We indicate the new C&C technique using the Ethereum blockchain, which is different from methods [1,2]. Existing malwares can easily use such technique, since the malwares using the method use HTTP/HTTPS protocol to get commands from botmaster.

2 Background

2.1 Ethereum

"Ethereum is a decentralized platform that runs smart contracts: applications that run exactly as programmed without any possibility of downtime, censorship, fraud or third-party interference."[1]

Two Types of Accounts. Ethereum has two types of accounts: Externally Owned Account (EOA) and Contract account. EOA is used to send Ether to another account, make contracts, and execute the contract. It is controlled by a

[1] Ethereum: https://www.ethereum.org/.

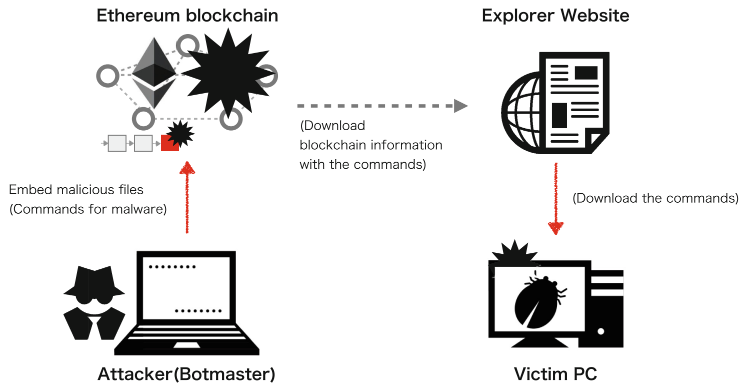
Ethereum blockchain
Explorer Website
(Download blockchain information with the commands)
Embed malicious files (Commands for malware)
(Download the commands)
Attacker(Botmaster)
Victim PC

Fig. 1. Threat of blockchain poisoning attack: When performing the poisoning attack, attackers embed malicious files into the targeted blockchain. We examined the actual situation of blockchain poisoning as stated in Sect. 5. In addition, when abusing the blockchain in C&C, botmasters embed commands into the blockchain, then the Explorer website downloads blockchain information with the commands. The malware obtains the commands by accessing the Explorer. We analyzed the feasibility of C&C technique with blockchain poisoning in Sect. 6.

private key[2]. In contrast, a Contract represents an account of smart contract on Ethereum itself. In other words, EOA sends a transaction to a Contract account to execute the contract.

Smart Contract and Flexible Space on Ethereum. Smart contract on Ethereum is written in bytecode, known as EVM code, and is executed on a virtual machine called Ethereum Virtual Machine (EVM), which runs on a node in the Ethereum network. Ethereum transactions have spaces that are used for the smart contract: *init* and *data* areas. *Init* area contains EVM code and is used to deploy the smart contract. In contrast, *data* area is used to call a function of a contract and give arguments to the function. This area can be freely used irrespective of smart contract.

Because both *init* and *data* areas are arrays of unlimited bytes (according to Ethereum yellow paper [22]), there are no theoretical upper limits to the size of data. However, because a transaction fee cannot exceed the block fee limit determined by miner voting, there is a practical upper limit for the data size. In addition, Go Ethereum, which is the official Go implementation of Ethereum, has another limitation. With its comment "Heuristic limit, reject transactions over 32 KB to prevent DOS attacks", a filter was added to the application to reject transactions containing more than 32 kB data in its *init*/*data* area. This limitation has been added since version v1.6.6. According to ethernodes.org[3], which

[2] Ethereum Development Tutorial https://github.com/ethereum/wiki/wiki/Ethereum-Development-Tutorial.
[3] ethernodes.org: https://www.ethernodes.org/.

contains information about the nodes in the Ethereum network, Go Ethereum runs on approximately 50% of all nodes in the Ethereum network. To the best of our knowledge, there is no limitation to the data size in other Ethereum client applications.

METAMASK. It is an Ethereum wallet implemented as an extension of web browsers: Chrome, Firefox, and Opera[4]. It supports connecting to not only Ethereum mainnet but also private net, and transferring ERC20 token.

Explorer. It is a web services that provides information about blockchain through web browsers without special software or running blockchain nodes. The services are important for transparency in blockchain.

Etherscan[5] is a well-known explorer in Ethereum. It provides information related to the Ethereum blockchain such as source code of contract, data in *init* and *data* areas, block hash, and transaction hash.

2.2 How to Use Smart Contract on Ethereum

We can use smart contract on Ethereum by the following two steps.

Deploy a Contract. We define a contract using high-level programming languages. A well-known language to define contracts on Ethereum is Solidity[6], which looks like JavaScript. The contract is compiled into byte-code called EVM code.

Next, the EVM code is deployed on the Ethereum network to use the contract. The input EVM code is returned by the compiler as output to *init* of a new transaction, without an Integrated Development Environment (IDE), the recipient address of the transaction is set to *null*. At this time, a contract account associated with the contract has not been generated yet. After the transaction is sent using the Ethereum wallet, it is broadcasted in the network and miners in the Ethereum network put the transactions into the new block to generate the next block. In the mining process, the Contract account is generated.

Execute the Contract. To execute a contract, a transaction must be made. You input byte-array to specify which function is called and give arguments to the function to *data* area for new transaction.

The transaction is then sent from EOA to the Contract account. In the mining process, the result of the contract execution is reflected in the blockchain.

Furthermore, in order to send a transaction containing EVM code, certain Ethereum wallet applications have functions that receive data in hex string and write it into a transaction.

---

[4] METAMASK: https://metamask.io.

[5] Etherscan: https://etherscan.io.

[6] Solidity: https://github.com/ethereum/solidity.

3 Related Work

Smart contract extends the function of blockchain and provides a programmable logic platform for all users on the blockchain network. However, when considering the security risks, it is reasonable to not allow programmable operations to untrusted users. So far, researchers have studied security and attacks regarding blockchain, crypto assets, and smart contract [9,13,21]. In this section, we introduce the attacks in programs and stored data related to smart contract and poisoning attacks related to crypto assets.

3.1 Data Stored Space Attack

Matzutt et al. studied the data inserted into the Bitcoin blockchain [15] and proposed countermeasures for undesirable data [14]. They discussed the benefits and risks of arbitrary blockchain contents and summarized some methods for data insertion into the Bitcoin blockchain. In addition, they found several types of undesirable content, including child pornography and violate another individual's privacy on the Bitcoin blockchain.

However, the spaces designed for data insertion on the Bitcoin blockchain are caused by OP_RETURN and Coinbase transaction and only the miner can insert data into Coinbase transaction. Hence, usual network participants can insert only 80 bytes of data to OP_RETURN at once. In contrast, Ethereum has more flexible space than Bitcoin. If a blockchain has designed flexible spaces, the poisoning attack becomes easier. Thus, we examine the possibility of the poisoning attack in Ethereum.

3.2 Programs Attack

Atzei et al. [3] reported a series of security vulnerabilities in Ethereum smart contracts. They categorized the vulnerabilities into three levels (Solidity, EVM, Blockchain) based on the causes of vulnerability and explained them by examining the source codes written in Solidity. The attacks work against the users and administrators of the smart contracts with vulnerabilities. In contrast, the poisoning attack, explained in this paper and [14,15], affects all users of a system which contains blockchain. This indicates that poisoning attack is a direct attack against blockchain.

3.3 C&C Technique Using Blockchain Network

Ali et al. [1,2] proposed C&C mechanism that leveraged the Bitcoin network. They indicated methods to insert C&C payload to Bitcoin transactions, except our method, which are reported in Sect. 3.1. Furthermore, they explained the advantages of C&C using Bitcoin network including the difficulty of takedown without causing any harmful effects to legitimate Bitcoin users and the cost for maintaining a C&C network. They also built a botnet on the Bitcoin main network and measured *response time*, which is the time period from when the botmaster issues an instruction and it is successfully received by the bot, to evaluate the method.

If a blockchain has a designed flexible space, attacker can embed C&C payload on the blockchain and bots can access the payload on the blockchain more easily using web service called Explorer.

4 Blockchain Poisoning Attack

4.1 What Is Blockchain Poisoning Attack

Blockchain poisoning attack is an attack against blockchain by embedding malicious or illegal files in the flexible space of blockchain. Attackers can force nodes in the blockchain network to download the files. This causes DoS attacks against blockchain. The attack target can be the blockchain and its users.

Attackers perform the attack as follows:

1. An attacker prepares a malicious or illegal file.
2. The attacker embeds the file into the flexible space of transaction, and broadcasts the transaction in the blockchain network.
3. The malicious file is embedded into the blockchain through the mining process and then shared among the network participants.

Files used for blockchain poisoning attack can be privacy information, malwares, and any illegal contents. Such files are also described in [15].

4.2 Why Blockchain Poisoning Attack Is Critical/Impact of Blockchain Poisoning Attack

The reasons why the attack is critical are as follows:

- Blockchain is shared among participants of blockchain P2P network.
- Transactions contained blockchain are hard to be modified or cancelled.

In a blockchain system, each full node needs to store block data synchronized by the P2P network. It means that attackers force the nodes to download and store malicious or illegal files by embedding such files into the blockchain. Certainly, nodes which don't store full blockchain data, like SPV nodes in the network, are not damaged directly by the attack as much as full nodes. However, since blockchain network is backed mainly by full nodes, the attack affects all users of the blockchain indirectly.

Immutability is one of the important features of blockchain. Because of this feature, transactions contained blockchain are hard to be modified or cancelled. Hence, blockchain poisoning attack is considered to be more persistent than conventional poisoning attacks, such as DNS cache poisoning, against public database, because repairing a poisoned blockchain without hard fork is not feasible. In contrast, once a transaction fee is paid, it is easy to send transaction with malicious data. It means that blockchain is heavily damaged by the attack while attackers can perform the attack in a low degree of dificulty.

4.3 Application of Blockchain Poisoning

C&C Using Blockchain. In a blockchain system, each node needs to store block data synchronized by the P2P network. Anyone can access the blockchain data by connecting to the network. These features can be used for C&C.

The mechanism of C&C using blockchain has already been presented by Ali et al. [1,2]. In this technique, bots are implemented by modifying bitcoin node software (freely available), and the bots connect to the bitcoin P2P network and use the network for C&C.

The methods that malware gets commands from bot master via blockchain. These methods are different from the ones described in previous studies [1,2].

- Get commands from blockchain on victim node.
- Access Explorer website to get command.

The first one can be used only when the victim server is running as a node of the blockchain network. Because each node synchronizes blockchain and the blockchain data is stored on itself, a malware can easily access the commands by reading the blockchain data, which are embedded into blockchain by bot master, data which stored on the server. Using this technique, malwares can hide their C&C communications in P2P communications of the blockchain network. For example, a malware aims at the nodes of crypto assets to steal their private keys and then, can get commands without direct communications with bot master and C&C server.

In the second method, the malware accesses the Explorer website to receive commands. It is difficult to detect and prevent this attack technique because most companies allow HTTP/HTTPS protocols and the content of HTTPS communication is encrypted.

Note that such attack techniques, including the method presented by Ali et al. [1,2], are also disadvantageous for attackers. Data in blockchain are unchangeable and unremovable. Once the malware is found by security researchers, they can get information about C&C from the blockchain and analyze it.

Hash Rate Decreasing of Blockchain/Price Manipulation of Crypto Assets. After performing blockchain poisoning, publishing information such as "The blockchain contains illegal files!!" can produce a negative impression on the users that uses the blockchain and they might leave the system. This is a DoS attack against blockchain because the hash rate of a network is very important for security in the blockchain system.

If a crypto asset encounters the attack mentioned above, its price can be declined. Thus, the attacker can reduce the price and benefit from the price difference between before and after attacking.

5 Evaluation of Flexible Space

We investigated the programmable space on the Ethereum main network (from 0 to 6,988,614 in block height) (July 30, 2015–December 31, 2018 UTC).

5.1 Methodology

We detected the transactions embedded files using the file carving method. This method is used to recover files from the unallocated spaces of a storage, for instance, in digital forensics. It can identify files embedded in unknown binary data by techniques such as searching file headers and using file structures [12]. We used Foremost[7] in this evaluation because it is used in general and open source program.

The procedure of our investigation is as follows:

1. Convert data extracted from a transaction from a byte-array to a binary file and then save the file.
2. Input the binary file to a file carving tool.
3. If the tool detects some files in the binary file, record the information of the transaction

Eighteen file types were used in this study for detection: jpg, gif, png, bmp, avi, exe, mpg, wav, riff, wmv, mov, pdf, ole, doc, zip, rar, html, and cpp.

In this evaluation, we did not cover files divided and embedded into the blockchain separately and encoded in some way. Attackers need certain burden (e.g., management of files, gas, etc.) to hide the embedded files or data on a blockchain by dividing or encoding.

5.2 Files Embedded in Transactions

Our investigation of data extracted from the transactions showed that 154 files were embedded in the Ethereum blockchain.

Figure 2 shows the file-types of the extracted files. As evident from Fig. 2, approximately 80% of extracted files were image files (jpg, png, and gif). Most of the image content were not problematic as they were group pictures and landscape. However, some pictures consisted of undesirable content. In addition, the pictures, appearing at first glance to be normal, may be malicious because they can violate the privacy of others, and be abused by steganography techniques [5].

We found three exe files in the Ethereum blockchain. The MD5 hashes of these three exe files are shown below;

(1) c9a31ea148232b201fe7cb7db5c75f5e
(2) c1e5dae72a51a7b7219346c4a360d867
(3) c9a31ea148232b201fe7cb7db5c75f5e

The two files are evidently the same. We inputted these hashes to VirusTotal to evaluate the files and concluded that the three exe files are malware because the analysis result of file (1) and file (3) indicated that their rates of detection by anti-virus software are 56/70 and the result of file (2) indicated that its rate of detection is 58/66. Moreover, according to a report [19], a malware called W32.Duqu has the same hash value as the file (1) and file (3).

[7] Foremost: http://foremost.sourceforge.net/.

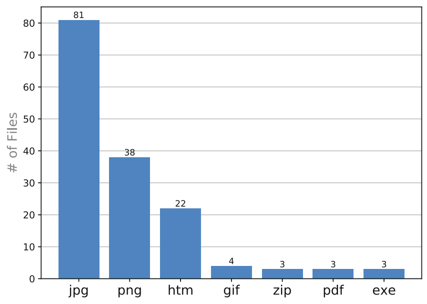

Fig. 2. The type of the embedded file

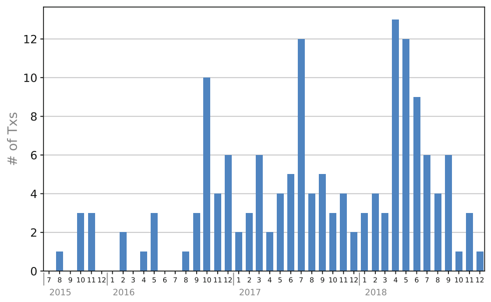

Fig. 3. Time-series histogram of file-embedded transactions

Table 1. Numbers of relation between sender and recipient accounts

(a) sender and recipient are the same	79
(b) sender and recipient are the different	70
(c) recipient is null (contract creation transaction)	5

Furthermore, we found that these three exe files were embedded by one account and the account sent three file-embedded transactions in approximately 6 min. Figure 3 shows time-series histogram of file-embedded transactions. We show when such transactions were contained by the Ethereum blockchain.

We demonstrate the relation the between sender and recipient of the transactions embedded files. The number of sender accounts was 113 and that of file-embedded transactions is 154. It indicates that some accounts embedded a file to the Ethereum blockchain several times. The maximum number of file-embedded transactions sent by one account is 10.

Table 1 presents the numbers of relation between sender and recipient accounts. There are three types of relations: (a) sender and recipient are the same, (b) sender and recipient are the different, and (c) recipient is null (i.e., contract creation transaction). Most accounts with file-embedded transactions send the transaction to any recipient except null. Some accounts embedded a file into a contract creation transaction.

6 Feasibility Experiment of Poisoning Attack

Determining the feasibility of an attack is important to assess the risk of the attack. We constructed an experimental environment, which imitates the actual environment of Ethereum blockchain, to assess the possibility of poisoning attack. In the environment, we attempted to embed files into our private Ethereum blockchain and extracted the same files from the Explorer using a web browser. We extracted and embedded the files to verify the usability of

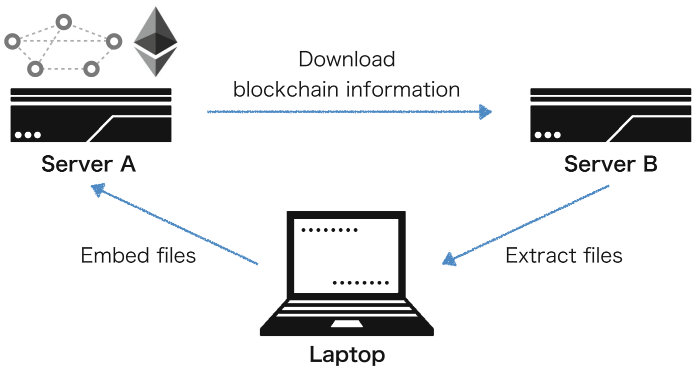

Fig. 4. Our experimental environment

blockchain poisoning as C&C infrastructure using web browsers and one-liner shell commands. This indicates ease of the attack.

6.1 Experimental Environment

Our experimental environment was constructed using two servers (Server A and Server B) and a laptop, as shown in Fig. 4. The components of the experimental environment are as follows:

Server A (Ethereum Privatenet). Go Ethereum is installed on this server that is configured to mine a private blockchain. This server plays the role of Ethereum mainnet in the actual environment.

Server B (Explorer). On this server, we set up a web server to display information about the Ethereum private blockchain on Server A using EthExplorer[8]. This web server plays the role of blockchain explorer, such as Etherscan in the actual environment.

Laptop. We used MacBook Pro to execute shell commands and Google Chrome (with METAMASK installed) for the experiment. These enabled the embedding and extracting of files in the same process as performing a poisoning attack against Ethereum main net.

6.2 Experiment

We explain the experiment procedure in three steps (preparation, embedding a file, and extracting the file). Because Go Ethereum rejects the transactions that contain data over 32 kB (as stated in Sect. 2.1), we used image files under 32 kB to embed into our private blockchain.

[8] EthExplorer: https://github.com/etherparty/explorer.

Preparation. This is the setting on Server A.

1. Start mining on Server A.
2. Connect METAMASK to Ethereum private net on Server A.
3. Create an account on METAMASK and send Ether to that account from the Coinbase account on Server A.
4. In the setting of METAMASK, turn on the "Show Hex Data" toggle switch.

Embed File. In this section, we describe the process to embed file on the blockchain.

1. To obtain the hexdump of a file (e.g., pic.jpg), the following command is executed and the result is copied to the clipboard.

   ```
   xxd -p pic.png | perl -pe 's/\ n//g'
   ```

2. Click "SEND" on METAMASK.
3. Select any account to send a transaction and paste the hexdump of the file to the field "Hex Data".
4. Click "NEXT" and "CONFIRM" to send transaction.
5. After the blockchain receives the transaction, we can confirm that the blockchain contains the hexdump in the web application on Server B.

Extract File. The process to extract a file from the blockchain is given below. It should be noted that the name of the extracted file is "pic_extracted.png".

1. Copy a hex string showed on the web application of Server B.
2. Replace <hex> by the following command on the hex string and execute it.

   ```
   echo <hex> | xxd -p -r > pic_extracted.png
   ```

6.3 Ease of Poisoning Attack

Section 6.2 demonstrates that files can be embedded into the Ethereum blockchain using only METAMASK and one-liner shell command. We used shell command to obtain the hexdump of files in this experiment, there is web service which provides conversion from binary file to a hex string. It is substantially possible to embed files into the Ethereum blockchain only using the web browser.

Section 6.2 demonstrates that the embedded data can be accessed using explorer website and extracted using a one-liner shell command.

In this experiment, we used a image file, which is about 28 KB, to embed. The transaction embedded the file costs 1952132 of gas.

7 Discussion

7.1 Behavior of a Suspicious Account

As stated in Sect. 5.2, we found that three exe files, judged as malware by VirusTotal, were embedded into the Ethereum blockchain by a single account.

Heuristic analysis, which detects malwares analyzing suspicious behavior, is a popular malware detection technique [11,18,23]. Considering the possibility that heuristic analysis can be implemented against blockchain poisoning, we observed the behavior of the suspicious account.

We found in our evaluation that the suspicious account sent and received ten transactions in total. These transactions are given below. Assume that "***account X***" denotes the suspicious account and *account A, B, and C* denotes the normal accounts.

(1) "An Ether transfer transaction" from *account A* to ***account X***
(2) "An Ether transfer transaction" from ***account X*** to *account B*
(3) ∼ (8) Self-sent transactions with data in their *data* area
  (3) 4.1 kB random-like data
  (4) 20.5 kB random-like data
  (5) broken PNG image
  (6) malware EXE file (1)
  (7) malware EXE file (2)
  (8) malware EXE file (3)
(9) "An Ether transfer transaction" from ***account X*** to *account C*
(10) Empty transaction from *account C* to ***account X***

This series of transactions indicates an attempt by an attacker to check if malicious files can be embedded into the blockchain. This attempt can be interpreted as follows.

First, this suspicious account received some Ether (1) required to send transactions and embed data into transactions. To check the data size that can be embedded into the blockchain, this account sent two transactions of different sizes (3)(4). Next, the account sent a transaction embedded PNG file to check if it can be embedded into the blockchain (5). The account then sent three transactions containing malware binary file to check if malicious files can be embedded into the blockchain (6)(7)(8).

We found suspicious behavior by a single account. However, there is a clear difference between malware and such suspicious accounts. Malwares have a series of actions. In contrast, attackers can also create unlimited accounts (say, address) on a blockchain. Although a single account sent the transactions in this case, they could be divided and sent by multiple accounts. Therefore, owing to the difference between malwares and accounts on blockchains, performing heuristic analysis on each account is ineffective and countermeasures against blockchain poisoning should be made by observing each transaction instead of account.

7.2 Risk of Flexible Space of Blockchain

We discuss the risks due to the flexible space of Ethereum blockchain, considering our evaluation of programmable space and our experiments on poisoning attack. Following are the reasons for increasing the risks.

Functions for Embedding Data Are Officially Provided. Techniques to embedding arbitrary data into the Bitcoin blockchain [15] are not officially allowed except for OP_RETURN and Coinbase transaction. Therefore, when embedding any data into the blockchain that does not have flexible space, it is necessary to use difficult way like "coding Bitcoin script" because general wallets of such blockchains do not have embedding functions.

However, in the case of blockchains that have flexible space, including Ethereum, some functions for writing data to the space are officially provided. As stated in Sect. 6, attackers can use wallet apps, which can be also used by benign users, to perform a poisoning attack through the same procedure. Therefore, a flexible space in blockchain facilitate poisoning attack.

Explorer Exists. As described in Sect. 2.1, there are websites called Explorer in each blockchain, which provide information related to the blockchain. The content of these websites are different according to the structure and functions of each blockchain. If a blockchain has flexible space, the embedded data can be easily obtained in any form such as ascii and hexadecimal.

Ali et al. [1,2] proposed the C&C technique using Bitcoin blockchain. As mentioned before, Explorer websites can be used in C&C on both Bitcoin and Ethereum. We can get data embedded into blockchain via explorer using a web browser, according to the experiment in Sect. 6. It means that it is possible to get embedded data via HTTP/HTTPS, whose communication is allowed in many companies. For an attacker using C&C, blockchain with flexible space is an advantage.

In the study [1,2], they built a botnet using the Bitcoin network by Bitcoin SPV client. As mentioned above, the technique that uses Explorer website can use HTTP/HTTPS to communicate with C&C server. In the past, malwares that use blogs and social media [6] and those that use GitHub [17] for C&C have been reported. The techniques that use Explorer can be easily implemented on such malwares.

It is important for attackers to not be involved in the migration of data from the blockchain to the Explorer website. As stated in Sect. 6, attackers access the blockchain network while embedding data into the blockchain. However, when bots receive the data, they independently access the Explorer website. This implies that C&C technique using blockchain Explorer is effective for an attack because it is difficult to link a bot to an attacker.

Listing 1.1. Example of contract with hexdump of a file

```
pragma solidity ^0.5.0;

contract Test {
    function testfunc() public pure returns(bytes memory){
        bytes memory data = hex"<hexdump of a file>";
        return data;
    }
}
```

7.3 Possibility of Wrapping Arbitrary Binary in a Contract

Unfavorable data can be embedded into blockchain owing to flexibility. Hence, as a simple countermeasure for unfavorable transactions in blockchain network, we can decrease the flexibility of the space used for embedding data.

Most files found in the studies described in Sect. 5 are embedded in form of hexdump. In Ethereum, transactions that have hexdump of files in their *init* area can be rejected by allowing only valid data as EVM code.

However, the source code can be successfully compiled into an EVM code even if it is embedded hexdump, like the Solidity source code showed in Listing 1.1. This implied that we can make a valid EVM code containing arbitrary hexdump of files. Therefore, it is difficult to take measures against embedding malicious data into a blockchain.

7.4 Countermeasure Against Blockchain Poisoning

As one of the countermeasures against content insertion, *mandatory minimal fees* to penalize large transactions is proposed [14]. Certainly, economic costs are effective because Proof of Work has never been broken except for some specific situations such as the decrease of hash rate. However, the proposed method [14] has a drawback due to a lack of correlation between the size and maliciousness of the data.

The following can be considered as a new naive countermeasure against blockchain poisoning attack. The countermeasure adopt mandatory minimal fee determined by its similarity with other contracts. The fee is calculated using all contracts which are deployed on the blockchain in the past and it becomes larger when there are no similar contracts. As the number of similar contracts increases, the minimal fee decreases.

In this mechanism, the cost for sending a transaction depends on the number of similar contracts instead of the size of transactions. Because the number of such malicious transactions is small in the blockchain, the attacker can be forced to pay large costs.

Previous study [10] demonstrated that there are similar contracts that are actually used owing to some reasons such as reusing source codes of contracts. In a blockchain network, the content of blocks are determined by consensus between the network participants. However, it is generally difficult to take a consensus if a file is benign or malicious. Therefore, we demonstrated the important approach to obtain consensus using the similarity of contracts as one of the solutions to the problem.

8 Conclusions

To assess the risk of blockchain poisoning attack, we analyzed the Ethereum blockchain to examine the actual situation, verified the ease of attack using our experimental blockchain environments.

We confirmed that 154 files were embedded on the Ethereum blockchain, including some malicious files. Furthermore, we showed that blockchain poisoning can be easily performed using web browsers and one-liner shell command. We indicated the possibility of C&C technique using the Ethereum blockchain, which is different from previous methods [1,2]. The method can be easily applied to existing malwares because they use HTTP/HTTPS protocol to receive commands from botmaster.

Acknowledgement. This work was partly supported by the Grant-in-Aid for Scientific Research (B) (19H04107).

References

1. Ali, S.T., McCorry, P., Lee, P.H.-J., Hao, F.: ZombieCoin: powering next-generation botnets with bitcoin. In: Brenner, M., Christin, N., Johnson, B., Rohloff, K. (eds.) FC 2015. LNCS, vol. 8976, pp. 34–48. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48051-9_3
2. Ali, S.T., McCorry, P., Lee, P.H.J., Hao, F.: ZombieCoin 2.0: managing next-generation botnets using bitcoin. Int. J. Inf. Secur. **17**(4), 411–422 (2018)
3. Atzei, N., Bartoletti, M., Cimoli, T.: A survey of attacks on ethereum smart contracts (SoK). In: Maffei, M., Ryan, M. (eds.) POST 2017. LNCS, vol. 10204, pp. 164–186. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-662-54455-6_8
4. Azaria, A., Ekblaw, A., Vieira, T., Lippman, A.: MedRec: using blockchain for medical data access and permission management. In: 2016 2nd International Conference on Open and Big Data (OBD), pp. 25–30. IEEE (2016)
5. Cheddad, A., Condell, J., Curran, K., Mc Kevitt, P.: Digital image steganography: survey and analysis of current methods. Sig. Process. **90**(3), 727–752 (2010)
6. Chen, J.: Blackgear cyberespionage campaign resurfaces, abuses social media for c&c communication (2018). https://blog.trendmicro.com/trendlabs-security-intelligence/blackgear-cyberespionage-campaign-resurfaces-abuses-social-media-for-cc-communication/. Accessed 13 Dec 2018
7. Dorri, A., Kanhere, S.S., Jurdak, R., Gauravaram, P.: Blockchain for IoT security and privacy: the case study of a smart home. In: 2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), pp. 618–623. IEEE (2017)

8. Fromknecht, C., Velicanu, D.: CertCoin: a NameCoin based decentralized authentication system 6. 857 class project (2014)
9. Hasanova, H., Baek, U., Shin, M.G., Cho, K., Kim, M.S.: A survey on blockchain cybersecurity vulnerabilities and possible countermeasures. Int. J. Netw. Manage. **29**(2), 2060 (2019)
10. Kiffer, L., Levin, D., Mislove, A.: Analyzing ethereum's contract topology. In: Proceedings of the Internet Measurement Conference 2018, pp. 494–499. ACM (2018)
11. Kolbitsch, C., Comparetti, P.M., Kruegel, C., Kirda, E., Zhou, X.y., Wang, X.: Effective and efficient malware detection at the end host. In: USENIX Security Symposium, vol. 4, pp. 351–366 (2009)
12. Laurenson, T.: Performance analysis of file carving tools. In: Janczewski, L.J., Wolfe, H.B., Shenoi, S. (eds.) SEC 2013. IAICT, vol. 405, pp. 419–433. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39218-4_31
13. Li, X., Jiang, P., Chen, T., Luo, X., Wen, Q.: A survey on the security of blockchain systems. Future Gener. Comput. Syst. (2017)
14. Matzutt, R., Henze, M., Ziegeldorf, J.H., Hiller, J., Wehrle, K.: Thwarting unwanted blockchain content insertion. In: 2018 IEEE International Conference on Cloud Engineering (IC2E), pp. 364–370, April 2018
15. Matzutt, R., et al.: A quantitative analysis of the impact of arbitrary blockchain content on bitcoin. In: Meiklejohn, S., Sako, K. (eds.) FC 2018. LNCS, vol. 10957, pp. 420–438. Springer, Heidelberg (2018). https://doi.org/10.1007/978-3-662-58387-6_23
16. Nakamoto, S., et al.: Bitcoin: A Peer-to-Peer Electronic Cash System (2008)
17. Pernet, C.: Winnti abuses GitHub for C&C communications (2017). https://blog.trendmicro.com/trendlabs-security-intelligence/winnti-abuses-github/. Accessed 13 Dec 2018
18. Song, D., et al.: BitBlaze: a new approach to computer security via binary analysis. In: Sekar, R., Pujari, A.K. (eds.) ICISS 2008. LNCS, vol. 5352, pp. 1–25. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-89862-7_1
19. Symantec: W32.duqu the precursor to the next stuxnet (2011). https://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_duqu_the_precursor_to_the_next_stuxnet.pdf
20. Szabo, N.: Smart contracts: building blocks for digital free markets. Extropy, no. 16 (1996). http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart_contracts_2.html
21. Wohrer, M., Zdun, U.: Smart contracts: security patterns in the ethereum ecosystem and solidity. In: 2018 International Workshop on Blockchain Oriented Software Engineering (IWBOSE), pp. 2–8. IEEE (2018)
22. Wood, G.: Ethereum: a secure decentralised generalised transaction ledger. Byzantium version (2018). https://ethereum.github.io/yellowpaper/paper.pdf. Accessed 3 Dec 2018
23. Yin, H., Song, D., Egele, M., Kruegel, C., Kirda, E.: Panorama: capturing system-wide information flow for malware detection and analysis. In: Proceedings of the 14th ACM Conference on Computer and Communications Security, pp. 116–127. ACM (2007)