Developing a K-ary malware using Blockchain

Joanna Moubarak
ESIB, USJ
CIMTI
Beirut, Lebanon
joanna.moubarak@net.usj.edu.lb

Maroun Chamoun
ESIB, USJ
CIMTI
Beirut, Lebanon
maroun.chamoun@usj.edu.lb

Eric Filiol
ESIEA
$(C+V)^O$ *Lab*
Laval, France
efiliol@netc.fr

***Abstract*—Cyberattacks are nowadays moving rapidly. They are customized, multi-vector, staged in multiple flows and targeted. Moreover, new hacking playgrounds appeared to reach mobile network, modern architectures and smart cities. For that purpose, malware use different entry points and plug-ins. In addition, they are now deploying several techniques for obfuscation, camouflage and analysis resistance. On the other hand, antiviral protections are positioning innovative approaches exposing malicious indicators and anomalies, revealing assumptions of the limitations of the anti-antiviral mechanisms. Primarily, this paper exposes a state of art in computer virology and then introduces a new concept to create undetectable malware based on the blockchain technology. It summarizes techniques adopted by malicious software to avoid functionalities implemented for viral detection and presents the implementation of new viral techniques that leverage the blockchain network.**

***Index Terms*—Malware, K-ary Virus, Malicious program, Blockchain, APT**

I. Introduction

Computer infections hit the mainstream in recent years exploiting systems vulnerabilities and creating specific malicious softwares that are penetrating organizations. There are several definitions of the concept of computer infection, but none is really complete as recent developments are not taken into account. In the other hand, in order to hinder analysis, remain undetected and persist in the network, typically malware use passive and active self-defense mechanisms [1] [2]. However, these adopted techniques have many limitations and they are obviously apprehended by most antiviral solutions. First, these mechanisms usually necessitate the combination of several techniques. Furthermore, they are difficult to implement and manage. Moreover, some of these techniques may modify the code by adding random instructions or delay the analysis but the final result is the same and the encryption procedures remain unchanged. In the rest of this paper, we will present a new approach of malware conception using blockchain and based on the k-ary concept.

This paper is the result of a prolonged survey on viral techniques adopted by malware as we go through several computer virology studies [3]. Also, some malware samples were examined in real time against several analysis approaches [4] and in the other hand, many antiviral solutions were tested in different attack scenarios and networks. Moreover, several blockchain architecture types [5] were considered while developing the new malware.

978-1-5386-3416-5/18/$31.00 © 2018IEEE

The remainder of this paper is organized as follows: Section II develops a summary of a k-ary malware, followed by the utilization of the blockchain potential to create a k-ary malware in Section III. We conclude this paper and present our future work in Section IV.

II. The K-ary malware

This section presents a new category of malware denoted k-ary malware. As an alternative of holding the whole instructions constituting a malicious program in one file, this category encompasses k separate chunks which constitute a partition of the full code. Each of these programs holds only a subdivision of the instructions and reflects a regular uninfected program.

A. k-ary malware definition

The K-ary malware was introduced initially in 2007 [6] and has been later validated by several proof-of-concepts (POCs) [7]. The formalization of this new type of malware is generalized from Cohen's model using another approach based on vector Boolean functions in order to study software interactions. The modalization has proved that simple and polymorphic/metamorphic infections are one way or another correspondent due to the fact that the full information is accessible after the first step of infection. Whereas, the interesting element in k-ary malware consists in the segregation of information.

Essentially, a k-ary malware is a combined virus where the viral payload is separated and distributed into k different files. Each part looks like an innocent executable file and do not generate any indication of compromise (IOC). Two main categories of k-ary codes exists [6]:

(i) Class I code: The k parts are working sequentially. Three subcategories are to be considered depending on the relation between the several parties. The execution of these k files can be dependent from the others files ($A \ \ subclass$), no part is denoting the other ($B \ \ subclass$) or semi-dependent from their execution ($C \ \ subclass$).

(ii) Class II code: The k parts are working in parallel. Thus, all chucks have to be available and active in the system at the same period.

Furthermore, the k-ary malware are represented in Van Wijngaarden grammars defining the selection of the malware parts [8]:

$$\alpha R_m \gamma \Leftrightarrow \{\exists \omega \in (\alpha \otimes \gamma) | \omega \in C(G_m)\}$$

If the result of the selection function $\otimes$ of two files α and γ is a part of the code C generated by the malware m then it is a k-ary code.

B. Complexity

While Cohen [3] and Adleman [9] analyzed viruses with analogy to Turing machines and recursive functions and a generalized model for malicious behavior have been defined in [10] and [11], these studies do not reflect new malware interactions. For instance, the formalization of combined viruses have been well studied in [12]. Furthermore, it has been demonstrated that the problem of detecting a k-ary malware is NP-complete [13] [14] and that the presence of all codes in memory in Class II codes and Class I (A and C subclasses) constitutes a flaw, except when using a joint rootkit technology. On the other hand, in [8], the automatic generation of K-ary codes have been detailed, sustaining their detection difficulties and their complexity. Also, in [15], k-ary codes were modulated through Join Calculus and have been demonstrated to be undecidable, except for a calculus fragment not inevitably applicable.

Therefore, given the NP-completeness of k-ary codes detection, in order to explore the feasibility of a truly undetectable malware, we will use the concept of combined viruses using the blockchain network.

C. Implementations

Multiple proof-of concepts confirmed the complexity of combined viruses for OpenOffice in win32 and Linux environment [14].

Moreover, the different subclasses were validated in serial ($4 \leq k \leq 8$) [16] and in parallel (k = 4) [16]. For instance, each part has been able to regenerate the missing codes under different nomenclatures.

Furthermore, a k-ary virus was implemented in Python [17] in order to share a secret key utilized to decipher the viral payload. The first use case randomly divided the key between the different parts. However, this method requires the availability of all parts in order to retrieve the payload. Another solution consisted in adopting Shamir's Secret Sharing with Neville/Aitken's algorithm to resolve the key and implement the k-ary virus.

III. The Blockchain potential in a k-ary malware

As stated previously, we will develop a k-ary malware utilizing Blockchain. This section gives an overview on blockchain networks and their features and exposes the new k-ary malware implementation and testing.

A. Blockchain Overview

The blockchain is a secure peer-to-peer environment used to maintain a public ledger of transactions between parties where trust is utilized to achieve consensus. This latter depends on several algorithms and typically differ according to the blockchain type and the Distributed Ledger Technology (DLT) employed. Primarly, the blockchain is an immutable data structure using blocks as memory units where each block is referenced by its hash. To characterize the transactions, the root of the Merkle tree is stored. Each block is composed by several transactions where digital signatures and cryptographic schemes are used to verify each transaction. Moreover, heterogeneous nodes are supported in the distributed network. Each node will verify and broadcast the block until reaching a consensus. The first miner to validate the blocks is rewarded. At the time of writing this paper, there exist more than 700 blockchain types and most of them are alternatives variants of the Bitcoin blockchain. We explored this technology by testing the three mains DLTs in the market nowadays namely Bitcoin, Ethereum and Hyperledger. Mainly, the difference between these networks comes from the fact that Bitcoin and Ethereum are permission-less networks whereas all parties needs to be identified in the Hyperledger blockchain. In addition, the concept of smart contracts, which are function codes compiled with valid transactions, only exists in the Ethereum and Hyperledger networks.

For a long time, the blockchain technology was associated with the Bitcoin DLT, based on the Proof-of-Work mechanism. However, each DLT network is characterized by its own features and consensus algorithm. Regardless of the DLT type, the blockchain technology offers numerous security features. Therefore, the building blocks offered a trusted platform that applications are build on top [18]. However, malicious entities took also advantages from this backbone. Cryptocurrencies theft and the 51% problem where a self-interested miner owns the majority of network work in a Proof-of-Work consensus (in Bitcoin and Ethereum early releases) are typical misuses. Moreover, cryptocurrencies are widely adopted by ransomware infiltrators. And in some cases, malicious contents are sold and uploaded to the blockchain encrypted and abused by the owners of the decryption key further than other mistreated scenarios and dark web applications. Furthermore, the Tor network recently leveraged the blockchain to conduct illegitimate activities [19]. Besides, in the next section, we will leverage the blockchain as a main entity to create the k-ary malware.

B. Malware design

As we have seen previously, several attacks scenarios leveraged the blockchain to conduct fraudulent activities. Whereas, for the conception of the new viral algorithms, the blockchain network is a crucial part of the new k-ary malware design.

Designing a k-ary malware will lead us to a key management problem to identify each node and a key generation problem to agree on the complexity of the keys [17]. Besides, we need to add randomization for more efficiency. To resolve these problems, we resorted to the blockchain technology.

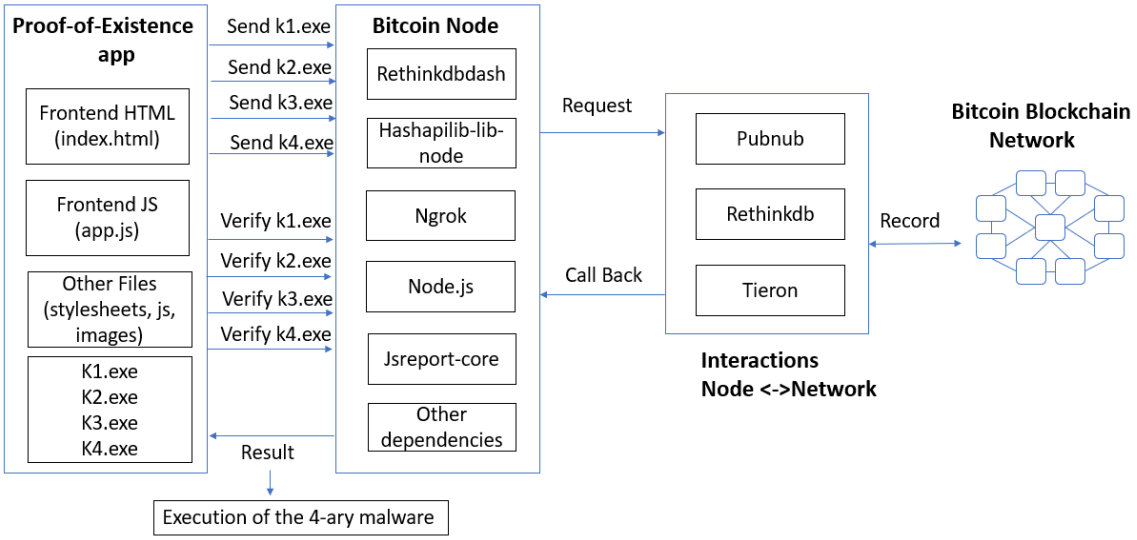

Fig. 1. 4-ary malware workflow

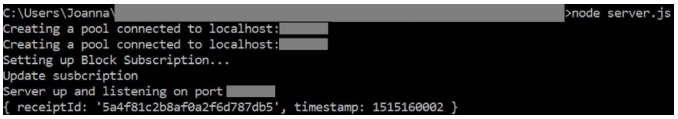

Fig. 2. Node.js

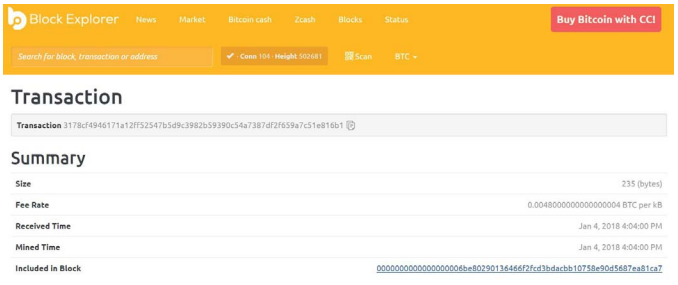

Fig. 3. k1.exe transaction summary.

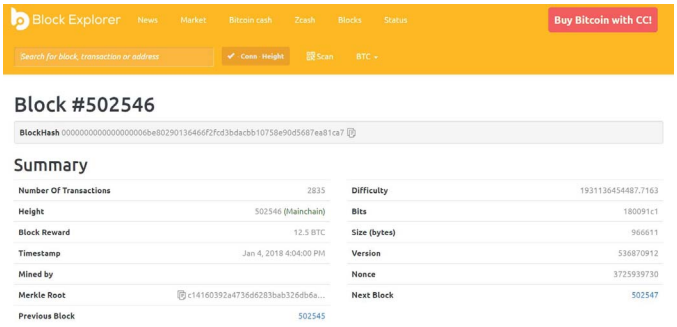

Fig. 4. k1.exe block summary.

In 1988, the authors of [20] proposed viruses as a solution for handling cryptographic keys. Besides, k-ary viruses are considered for this use case as well where the encrypted payload is confined in one part and the secret key available in another part. As for the proposed new viral algorithms, we have leveraged the cryptographic schemes, the hashing functions, and the digital signatures, which characterize the blockchain network, to develop the new k-ary malware. Fig. 1 shows the malware workflow.

The key components of the malware include a proof of existence application that interact with the Bitcoin Blockchain Network (see Fig. 2) through the integration of Pubnub, Rethinkdb and Tieron platforms. A detailed tutorial for this combination is given in [21] to which we referred to in the implementation. For the new k-ary algorithmic, the viral payload is splitted in 4 different files. The first viral mechanism employed is the auto-reproduction property generated by k1.exe. The second k-ary file include a keylogging action. The third executable file encompass the property to hide a specific file and the final k-ary executable permits the auto-execution at system startup. Alternatives or additional malicious activities can also be used. For example, the need to interact with a command-and-control server may also be written in a segregated file and added to the design. Also, the auto-deletion propriety implementation provides an interesting feature. Furthermore, breaking up more the code can add more stealthiness. For this POC, a 4-ary malware was tested. The next step consists in submitting the content to the blockchain which store a record of each file that anyone can verify its existence at any time in the blockchain explorer (see Fig. 3 and Fig. 4). Moreover, the hashing of each executable and the receipt that were given by the blockchain, will be recorded in Rethinkdb [1] through Tieron[2] APIs and Pubnub[3] real-time processing. Furthermore, the ngrok service provides a secure tunnel to connect with Tieron and receives

[1]An open-source database with real time capabilities.
[2]A helper platform to manage blockchain requests.
[3]A data stream network.

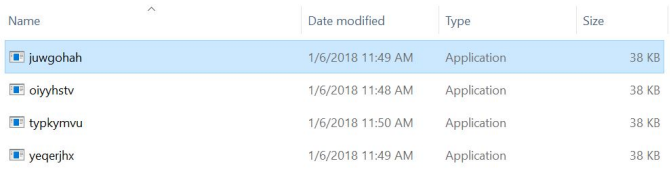

Fig. 5. k1 Execution: Auto-reproduction

callbacks.

In order to execute the 4-ary malware, we verified the existence of each file in the bitcoin network through the hashing signatures and if confirmed, we execute the viral payload (see Fig. 5). In the testing scenario, we created a class I independent 4-ary malware which is the most complex class in term of detection because no executable helps to spot the other.

C. Attack

Our proof-of-concept is based mainly on the proof-of-existence[4] application. Therefore, in order to convict other systems, a medium to interact with our malicious application is needed. Phishing or other techniques can be employed for that purpose. Besides, the core application databases and flows subscriptions are scheduled for some period of time after which the accounts utilized are removed. This makes the attacker anonymous.

IV. Conclusion

In 1986, F. Cohen proved that the detection of viruses is an undecidable problem. Besides, many defense strategies are deployed nowadays to fight against malware. Primary, network traffic analysis is used to create a baseline for ordinary network flows and spot anomalies. Also, full-packet capture mechanisms are deployed for better visibility, reporting and network forensics. Furthermore, payload and behavioral analysis in sandboxes are considered for advanced malware discovery. Moreover, application containment approaches are employed through agents to exclude potential offensive programs in containers and intercept their malicious activity. Finally, many agents are used for data collection and endpoints monitoring using intelligence to provide efficient protection and incident response.

According to the theory of computability, some problems are not calculable and the problem of viral detection is one of the undecidable problems. In this paper, we utilized the Blockchain in order to explore the feasiblity of a new undetectable malware. We based our malware on k-ary codes which have been demonstrated to be NP-complete. We have developed a 4-ary malware and tested it in real time where each chunk of the code interacts with the Bitcoin network to be validated and to make sure that it belongs to our malicious software. Therefore, the blockchain network provided an elegant solution to retrieve the multiple parts of the malware, making sure of their authenticity and integrity without worrying about the generation, the management and the storage of the keys.

The next step consists in leveraging smart contracts functions to enhance our k-ary malware and add more complexity. Besides, we will tackle its formalization and validate it in real time against advanced and sophisticated endpoints protections.

[4]An application where users can verify the existence of a particular content on the blockchain

References

[1] E. Filiol, "Malicious cryptology and mathematics," in *Cryptography and Security in Computing*. Intech, 2012.

[2] I. You and K. Yim, "Malware obfuscation techniques: A brief survey," in *Broadband, Wireless Computing, Communication and Applications (BWCCA), 2010 International Conference on*. IEEE, 2010, pp. 297–300.

[3] F. Cohen, "Computer viruses: theory and experiments," *Computers & security*, vol. 6, no. 1, pp. 22–35, 1987.

[4] J. Moubarak, M. Chamoun, and E. Filiol, "Comparative study of recent mea malware phylogeny," in *Computer and Communication Systems (ICCCS), 2017 2nd International Conference on*. IEEE, 2017, pp. 16–20.

[5] J. Moubarak, E. Filiol, and M. Chamoun, "Comparative analysis of blockchain technologies and tor network: Two faces of the same reality?" in *CSnet , 1st Cyber Security in Networking Conference*. IEEE, 2017.

[6] E. Filiol, "Formalisation and implementation aspects of k-ary (malicious) codes," *Journal in Computer Virology*, vol. 3, no. 2, pp. 75–86, 2007.

[7] M. Dalla Preda and C. Di Giusto, "Hunting distributed malware with the κ-calculus," in *Fundamentals of Computation Theory*. Springer, 2011, pp. 102–113.

[8] G. Gueguen, "Van wijngaarden grammars, metamorphism and k-ary malwares," *arXiv preprint arXiv:1009.4012*, 2010.

[9] L. M. Adleman, "An abstract theory of computer viruses," *Advances in Crypto*, 1998.

[10] Z. Zuo and M. Zhou, "Some further theoretical results about computer viruses," *The computer journal*, vol. 47, no. 6, pp. 627–633, 2004.

[11] G. Bonfante, M. Kaczmarek, and J.-Y. Marion, "On abstract computer virology from a recursion theoretic perspective," *Journal in computer virology*, vol. 1, no. 3, pp. 45–54, 2006.

[12] E. Filiol, "Metamorphism, formal grammars and undecidable code mutation," *International Journal of Computer Science*, vol. 2, no. 1, pp. 70–75, 2007.

[13] Filiol, "Malware of the future," 2015.

[14] D. de Drézigué, J.-P. Fizaine, and N. Hansma, "In-depth analysis of the viral threats with openoffice. org documents," *Journal in Computer Virology*, vol. 2, no. 3, pp. 187–210, 2006.

[15] G. Jacob, E. Filiol, and H. Debar, "Formalization of viruses and malware through process algebras," in *Availability, Reliability, and Security, 2010. ARES'10 International Conference on*. IEEE, 2010, pp. 597–602.

[16] É. Filiol, *Techniques virales avancées*. Springer, 2007.

[17] A. Desnos, "Implementation of k-ary viruses in python," *Hack. lu*, 2009.

[18] A. Bahga and V. Madisetti, "Blockchain applications: A hands-on approach," 2017.

[19] BlockchainBlog, "Blockchains and the Internet of Things," https://blog.blockchain.com/tag/tor/, 2017, [Online; accessed 12-November-2017].

[20] J. Riordan and B. Schneier, "Environmental key generation towards clueless agents," *Mobile agents and security*, vol. 1419, pp. 15–24, 1998.

[21] Pubnub, "Build a Proof of Existence Service in the Blockchain." 2017.