Forensic Analysis of Ransomware Families using Static and Dynamic Analysis

Kul Prasad Subedi, Daya Ram Budhathoki, Dipankar Dasgupta
Department of Computer Science
University of Memphis
Memphis, TN 38152, USA
Email: {kpsubedi,dbdhthki,ddasgupt}@memphis.edu

***Abstract*—Forensic analysis of executables or binary files is the common practice of detecting malware characteristics. Reverse engineering is performed on executables at different levels such as raw binaries, assembly codes, libraries, and function calls to better analyze and interpret the purpose of malware code segments.**
In this work, we applied data-mining techniques to correlate multi-level code components (derived from reverse engineering process) for finding unique association rules to identify ransomware families. However a reverse process and analysis of code structure do not always provide run-time behavior of executables so we used a combined approaches (*static* and *dynamic*) to better unveil hidden intent of the program. We performed analysis of 450 samples of ransomware and experimental results reported some important correlation among different code components from our combined analysis.

***Keywords*—Forensic, Malware, Reverse Engineering, Ransomware, Data-mining**

I. Introduction

Ransomware attack uses many tricks of Social Engineering (to get into the system), Virus properties (propagation, triggering and execution), cryptographic techniques (to lock system), use remote command and control (C&C) channel and crypto-currencies. In addition, ransomware attack exploits system weakness such as SMB vulnerabilities (MS-17-010) to get into the system. Ransomware generally encrypts the data in a victim's computer and ask for ransom to get the decryption key. Cyber criminals are using ransomware attacks frequently because it is very easy to make quick money. Moreover, the monetary transactions performed remained intractable due to the use of crypto-currencies like bitcoin. Each year new ransomware are released with advanced exploit techniques and attack vectors. Recently in May 2017, a widespread ransomware campaign was launched affecting as many as 150 countries including the United States, United Kingdom, Spain, Russia, France and Japan. The latest version of this ransomware is named as WannaCry, WCry or WannaDecryptor and requested a ransom amount of 0.1781 bitcoins roughly $300 US dollars [1]. Another variant of ransomware is named as SamSam and impacted multiple industries including Healthcare, Government and requested a total of roughly $325K US dollars ransom [4]. Similar to malware, ransomware utilizes all types of means (e.g., spam emails, mal-advertisements, social engineering) to propagate to a victim's computing system. Then, it will either lock the victim's system (i.e., **locker ransomware**) or encrypt the data (i.e., **crypto-ransomware**) in the victim's systems. Finally, it will require the victim to pay the ransom money in order to unlock the system or obtain the key for decrypting the data. According to the recent Internet Security Threat Report [2], the crypto-ransomware has now dominated the ransomware family. Therefore in this work, we mainly focused on static analysis assisted with reverse engineering of crypto-ransomware families. Examples of typical crypto-ransomware includes CryptoWall [11], CryptoLocker [16], and Locky [10], and typical locker ransomware includes Winlocker [21]. In comparison, the crypto-ransomware is much more harmful than the locker ransomware, since locker ransomware only locks the victim system, and the victim can still have access to his/her data by removing the storage from the infected system to an uninfected machine. Crypto-ransomware uses cryptographic encryption algorithms to encrypt the victim's data and the key may be stored in a remote C&C server, rendering it difficult to recover the data without paying the ransom money.

Most of the work on ransomware focuses on dynamic analysis using sandbox, a technique for running an untrusted program in safe environment without causing real harm to the system. Dynamic analysis has several drawbacks; for example, ransomware may detect the sandbox environment and may not execute in the sandboxing environment. Additionally, we are unaware of the arguments required for some command line malware unless we performed a thorough analysis. We claim that static analysis and reverse engineering is required to have through understanding of ransomware functionalities.

Figure 1 depicts the components of crypto-ransomware. These components are specific to the ransomware execution sequences. In Section VI, we described our methodology to discover assembly instructions, libraries, and function calls present in the code using *objdump* and *pe-parser*. In addition to these components, the *propagation strategy* is the first step used by the ransomware to get into the victim's machine. There are different methods being used to propagate as shown in Table I on the next page.

Table I depicts different ransomware families based on the following attributes: *Propagation Strategy*, *Date*

TABLE I: The list of the ransomware families. different families, propagation strategies, date appeared, cryptographic techniques used to encrypt data, and command and control (C and C) methods.

FAMILIES	Propagation Strategy	Date Appeared	Cryptographic Techniques	C and C Server
REVETON	Accused of illegal activities	2012	RSA and DES	Using MoneyPak
GPCODE	Email Attachments	2013	660-bit RSA and AES	Tor Network
CRYPTOLOCKER	compromised websites and email attachments	2013	2048-bit RSA	Tor Network
CRYPTOWALL	compromised websites and email attachments	2013	2048-bit RSA	Tor Network
FILECRYPTO	compromised websites and email attachments	2013	2048-bit RSA	Tor Network
TELSACRYPT	compromised websites and email attachments	2013	2048-bit RSA	Tor Network
CTB-LOCKER	Email Attachments	2014	Elliptic Curve Cryptography	Onion Network
CRYPTOMIX	Spear-phishing Email	2014	2048-RSA and AES-256 and ROT-13	P-2-P Network
CERBER	compromised websites and email attachments	2013	2048-bit RSA and RC4	Hardcoded IP range
PETYA	Link in an Email purporting to be a job application	2016	Elliptic Curve Cryptography and Salsa	Tor Network
SATANA	Email Attachments	2016	256-bit AES in ECB	Hardcoded IP Address
JIGSAW	Word Document with Javascript	2016	RSA and AES	Onion Network
SHADE	Spam Email	2015	RSA-3072 and AES-256	Fixed Server as C and C server
WANNACRY	Samba Vulnerability	2017	RSA and AES combination	Onion Network

Appeared, *Cryptographic Techniques*, and *Command and Control Server*. These attributes are very crucial and useful to in-place multi-layer defense strategies against ransomware attacks.

II. RELATED WORK

Several works have been done in Malware Structural Analysis using Reversing Engineering and Static Analysis. [7] presents the analysis of malware performing assembly code analysis to identify and classify the malware. Another work focused on malware using PE file structure analysis [28]. Taxonomy based [6] present an approach for preventing and detecting ransomware. The focus of our work is to monitor the encryption process of the victim's data.
Mercaldo et al. [17] used a static analysis method on Android system to automatically process ransomware sample. They performed with the goal of observing the malicious behavior. Modern techniques such as Software Defined Networking (SDN) have also be used to detect and mitigate the ransomware. [9] [8] Another technique such as honeypot based detection [18] was also used. In addition, recovery technique such as [25] has been done to defend against ransomware attack. There are approaches to deal with ransomware such as *File Hashes (Full or Portion)*, *Byte Signatures*, *System Behavior*, and *Network Signatures* [23]. These approaches are used by different antivirus engines, security tools, intrusion detection systems etc. The main problem with these approaches is that it is very easy to perform evasion. For example, File hashes can easily be bypassed by changing the equivalent assembly instructions. Similarly, Byte Signatures and network signatures are also prone to the same problem. System behavior is a unique approach but it requires the run time behavior of ransomware which is very difficult to capture.

III. PROPOSED METHODOLOGY

We used an integrated approach which combines static analysis (incorporating a data-mining technique) and run-time analysis to correlate code segments with dynamic behavior of ransomware.

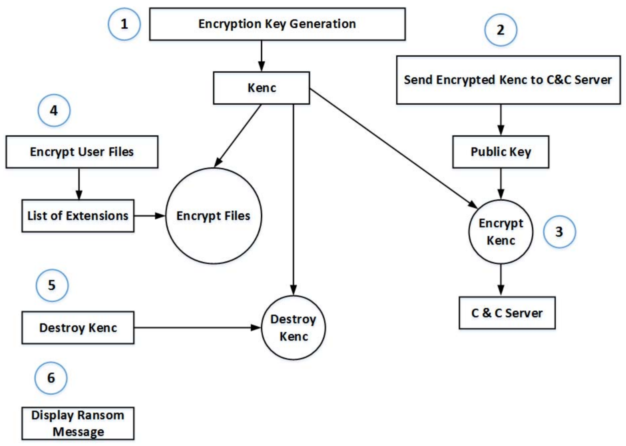

Fig. 1: Components of typical Crypto Ransomware where steps of execution are labelled with sequence numbers.

We developed an unique approach which perform static analysis of ransomware in three different levels: *Assembly*, *Library*, and *Function calls*. These three levels are able to capture ransomware behaviors in-terms of equivalent assembly instructions, function calls, and network signatures those are being used. We then applied data-mining technique to find association among components. The detail of these three contexts is described in Section IV. We then ran dynamic analysis of ransomware samples to validated the evolved association rules.

IV. DETAILS OF THE PROPOSED METHODOLOGY

Our approach first uses the static analysis of the code after performing reverse engineering. We design and implement the framework which contains reverse engineering and static analysis components. Reverse engineering is performed at three different levels. We apply reverse engineering based on assembly instruction level, libraries used in PE file structure level, and function calls used in the libraries. As shown in Figure 3, these libraries are specific to the functionality used by the ransomware. Ransomware uses a distinct set of libraries, unlike regular applications. This unique set of libraries can be considered the signature of the crypto-ransomware.

A. Reverse Engineering

Reverse engineering [13] is defined as the process which takes program binary file as input and produce the output in higher-level which is easier to understand. We leverage the existing disassembler *objdump* [24] to extract the assembly instructions from the binary files. There are two types of syntax normally used in assembly code: *Intel* and *AT&T*. We used the *Intel* syntax to disassemble the binary file. The output of the disassembler is the different codes used by the ransomware and we used preprocessor module to generate the frequency distribution. This frequency distribution is used to generate the vector representation of each ransomware.

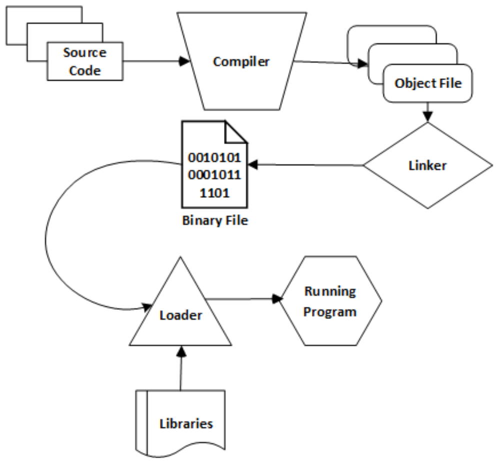

Fig. 2: Binary File Life Cycle

B. Portable Executable (PE) Format

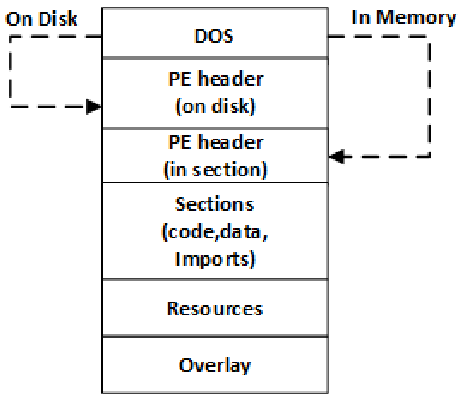

Fig. 3: Portable Executable (PE) File Format.

In the windows operating system, there are different types of binary files which follow PE format. These binary files are *executable files*, *dynamic link libraries*, *system files*, *activex controls*, *control panels*, and *screen savers*. All of these files have common PE structure. The portable executable file specification is described elsewhere [19]. PE file contains following sections as shown in Figure 3. PE file contains *headers* and *sections*. *Header* of the PE file contains: *DOS header*, *PE header*, *optional header*, *data directories*, and *sections table*. *Sections* contain: *code*, *imports*, and *data*. We designed and developed a tool to extract code segments which contain all DLLs required to execute the PE file. We are interested to find all such the DLLs and categorize them according to functionality based on crypto-ransomware components as shown in Figure 1. We used *objdump* [22] tool available in Linux which takes PE file as input and returns the corresponding assembly instructions. We leveraged the *objdump* program which is known as *assemblydumper*. We performed the analysis of forty-three samples of crypto-ransomware using *assemblydumper* which performs the parsing of the output of the *objdump* program and builds the frequency table of all the assembly instructions.

C. Static Analysis

Static analysis [26] is the code analysis without running an application. We designed and implemented the CRSTATIC (Crypt-Ransomware STATIC) tool as shown in Figure 4 which takes the binary program as input and generates the libraries used. These libraries are DLLs from the operating system. We used data-mining techniques to create the association rules of these DLLs used by ransomware. We performed the analysis of forty-three crypto-ransomware samples out of four hundred fifty malwares using our CRSTATIC. The results of this analysis is shown in Section VI.

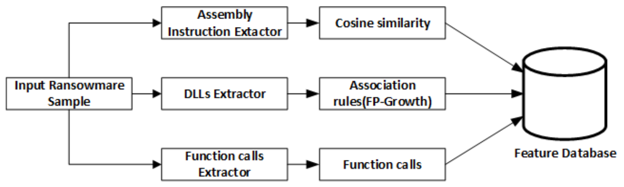

Fig. 4: Static Analysis of Ransomware to extract hidden features (CRSTATIC).

D. Association rule mining

Association rules are generally used to find the relations between items. Two of the most commonly used algorithms are Apriori and FP-Growth. The Apriori algorithm scans databases in each iteration so its slow and takes a lot of time. The FP-Growth algorithm scans [14] databases two times only and develops a tree by using the divide and conquer strategy,so it is fast and effective for the static analysis of ransomware. A list of DLLs from advanced static reverse engineering is provided as input to the FP-Growth algorithm. It produces a list of association rules as output. These rules are used as one of the signatures to detect ransomware.

Association rules consists of two parts in the form of if(antecedent) and then(consequent) and is represented in the form of expression such as X $\longrightarrow$ Y which means that whenever X is true Y also tends to be true. These Boolean rules are used to find a pattern to detect the ransomware. These association rules are used to build signatures to detect ransomware.

E. Cosine Similarity

Cosine similarity measures the similarity between two non-zero vectors. The cosine similarity score is used in positive space, where the score is ranged between zero and one. Each ransomware binary is represented in vector form. The vector is represented using Equation 1.

$$v = f_1 x_1 + f_2 x_2 + ... + f_n x_n \tag{1}$$

Where f represents the normalized frequency of the x instruction. We measure the similarity between normal binary as shown in Figure 6. Likewise, we also measure the similarity between ransomware as shown in Figure 5.

V. Malware Samples and Experiments

We have implemented a prototype CRSTATIC, which takes input as a PE binary file. The output of the CRSTATIC is further processed by a pre-processor module before applying a data-mining technique. CRSTATIC is implemented using C++ in a GCC compiler. We set up a test-bed for our experimental evaluation, using the following system configuration: Intel(R) Xenon(R) CPU X5550@2.67GHz, 6GB RAM, and 1.8 TB disk space. Our prototype implementation CRSTATIC is evaluated using following configuration and the dataset mention in below.

We collected four-hundred fifty malware samples from different sources, such as Virus Total, virushare, and from different research projects such as Shieldfs [12] and open source malware repository theZoo [5].

We performed the initial scanning of the malware carried out using RESTful API exposed by VirusTotal [27]. RESTful API takes two parameters: *apikey* and *resource*. The *apikey* is specific to the user account and *resource* is the hash value of the malware. However, RESTful API allows four requests to be made per minute. We used *time* module from Python to limit the four requests per minute. The response from VirusTotal [27] contains result after scanning the provided *resource* by using various ant-virus engines. Based on these results we have categorized crytpo-ransomware as shown in Table II.

VI. Analysis of Results

Table I shows different ransomware families based on the following attributes: *Propagation Strategy*, *Date Appeared*, *Cryptographic Techniques*, and *Command and Control Server*. These attributes are very crucial and useful to in-place multi-layer defense strategies against ransomware attacks. The propagation strategy is the initial phase of the ransomware attack. It defines how it leverages a range of possibilities including social engineering to specific vulnerability present in unpatched software installed in the target host to get access. For example, there are eight different propagation strategies as shown in the second column in Table I. *Compromised websites and email attachment* are the more frequently used propagation strategy. Propagation strategies provide what types of measures should be implemented to reduce the risk of the ransomware attacks which includes security awareness training, secure coding practice, secure software development life cycle, penetration testing etc. In addition, incident response team must be vigilant for further analysis of suspicious email attachments, compromised web site visits, emails, unpatched applications etc. The third column, which is entitled *date appeared*, shows the year when ransomware becomes wild. The fourth column shows the *cryptographic techniques* used by the ransomware to encrypt the target data. The cryptographic techniques used by the ransomware is a very crucial attribute to determine whether it is possible to decrypt data without paying the ransom. At a very high level, there are two approaches used by ransomware: *asymmetric* and *symmetric* algorithms. RSA is the commonly used *asymmetric* algorithm among most of the ransomware families. Similarly, AES is the commonly used *symmetric* algorithm are used in different families of ransomware which plays a key role to determine whether data can be decrypted without a key or not. For example, electronic code book (ECB) mode is leaks information regarding messages [20].

TABLE II: Crypto-Ransomware Families

Crypto Ransomware	
Family Name	Sample Size
Locky	138
CryptoWall	5
FileCryptor	11
TelsaCrypt	47
Crypt	2
CryptoLocker	2
Cerber	2
CTB_Locker	1
Petya	1
Satana	1
WannaCry	1

The following sections describe our approach in details. These approaches are: *Cosine Similarity Analysis of Assembly Code*, *DLLs loaded in crypto-ransomware*, and *Function calls specific to crypto-ransomware*.

A. Cosine Similarity Analysis of Assembly Code

Our dataset contains PE files which are the binary file formats used in the windows operating system. These PE files contain different sections as mentioned in Section IV. Our focus is in the code section of the PE file. We extracted corresponding assembly code from the code sections and calculated the frequency distribution of instructions. We have leveraged a built-in program in Linux: *objdump*. There are different types of assembly languages: *Intel*, *ARM*, and *MIPS*. We are using *Intel* format for cosine-similarity analysis. Figure 5 shows the similarity measure between ransomware families. The x-axis shows the vector elements which are assembly instructions and the y-axis shows the cosine value.

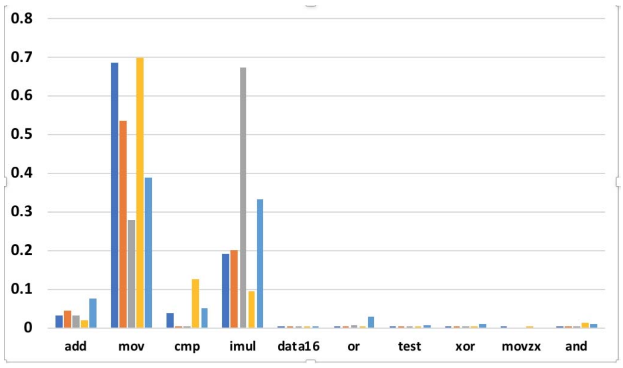

Fig. 5: Cosine Similarity between Ransomware Families.

Figure 6 shows the similarity measured between normal programs. The x-axis shows the vector elements which are assembly instructions and the y-axis shows the cosine value.

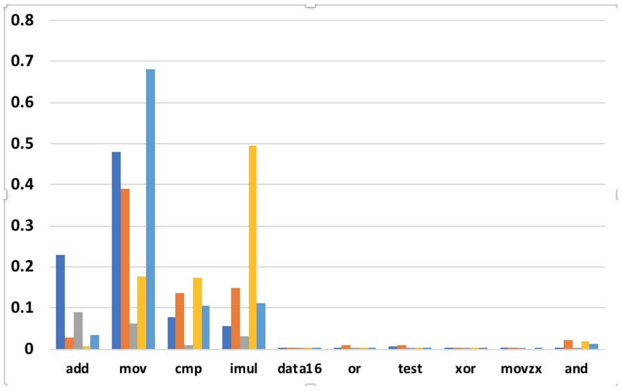

Fig. 6: Cosine Similarity between Normal Programs.

As shown in Figure 6 and Figure 5, we observe that the similarity score for *mov* instruction is less than 0.1 in the ransomware sample but more than 0.2 in normal binary. Similarly, for other instructions we observe the unique score between normal binary and ransomware binary.

B. DLLs loaded in Crypto-Ransomware

Multiple libraries are required to be loaded before any ransomware can be executed in the victim's Operating System. These libraries are called dynamic link libraries (DLLs) [15]. Ransomware uses a number of DLLs to provide the functionality such as connecting to the command and control server, generating the key, encrypting the files and destroying the key etc. The operating system loader is responsible to execute the ransomware binaries and maps the address of DLLs used in the ransomware binary to memory.

In order to extract the DLLs used by the ransomware binary, we leverage the *pe-parse* [3] which extracts the DLLs from the section header of the ransomware binary (PE file). Multiple DLLs specific to cryptography function calls are extracted as shown in TableIII. These DLLs are used to represent a ransomware binary in vector representation. Association rules based on FP-growth are applied to generate the list of rules as shown in Table IV.

TABLE III: Function calls in DLLs used in Ransomware

Name of DLL	Functions
ADVAPI32	CryptReleaseContext
	CryptAcquireContextA
	CryptGenRandom
	CryptEncrypt
	CryptGetKeyParam
	CryptAcquireContextW
	CryptDestroyKey
	CryptCreateHash
	CryptHashData
	CryptDestroyHash
	CryptGetHashParam
	CryptReleaseContext
	CryptSetKeyParam
	CryptImportKey
CRYPT32	CryptQueryObject
	CertFreeCertificateContext
	CertFindCertificateInStore
	CryptMsgGetParam
	CryptDecodeObjectEx
	CryptImportPublicKeyInfo
	CryptBinaryToStringA
	CryptStringToBinaryA
	CertGetNameStringW
	CertCloseStore
	CertFreeCertificateContext
CRYPTNET	CryptGetObjectUrl
CRYPTUI	CryptUIDlgSelectCertificateFromStore

Table III shows the list of DLLs used by the crypto-ransomware and the list of function calls used by the corresponding DLLs. For example, DLL library *ADVAPI32* has fourteen function calls which perform the encryption operation. Similarly, *CRYPT32* DLL library is used by the ransomware to carried out the encryption function as well.

TABLE IV: Association Rule on DLLs with score=1.0

Association Rules
[COMCTL32.DLL, SHELL32.DLL, USER32.DLL] ⇒ [KERNEL32.DLL]
[SHLWAPI.DLL, OLE32.DLL, ADVAPI32.DLL, SHELL32.DLL, USER32.DLL] ⇒ [KERNEL32.DLL]
[COMCTL32.DLL, ADVAPI32.DLL, SHELL32.DLL, USER32.DLL] ⇒ [KERNEL32.DLL]
[SHLWAPI.DLL, OLE32.DLL, ADVAPI32.DLL, SHELL32.DLL, USER32.DLL] ⇒ [KERNEL32.DLL]
[SHLWAPI.DLL, OLE32.DLL, ADVAPI32.DLL, SHELL32.DLL] ⇒ [KERNEL32.DLL]
[GDI32.DLL, ADVAPI32.DLL, SHELL32.DLL] ⇒ [KERNEL32.DLL]
[COMDLG32.DLL, ADVAPI32.DLL, USER32.DLL] ⇒ [KERNEL32.DLL]
[COMCTL32.DLL, ADVAPI32.DLL, USER32.DLL] ⇒ [KERNEL32.DLL]
[MPR.DLL, ADVAPI32.DLL, SHELL32.DLL, USER32.DLL] ⇒ [KERNEL32.DLL]
[VERSION.DLL, SHELL32.DLL, USER32.DLL] ⇒ [KERNEL32.DLL]
[SHLWAPI.DLL, SHELL32.DLL, USER32.DLL] ⇒ [KERNEL32.DLL]
[COMDLG32.DLL, SHELL32.DLL, USER32.DLL] ⇒ [KERNEL32.DLL]
[OLE32.DLL, ADVAPI32.DLL, SHELL32.DLL, USER32.DLL] ⇒ [KERNEL32.DLL]
[COMDLG32.DLL, ADVAPI32.DLL, SHELL32.DLL, USER32.DLL] ⇒ [KERNEL32.DLL]
[MPR.DLL, VERSION.DLL] ⇒ [ADVAPI32.DLL]
[COMCTL32.DLL, SHELL32.DLL, KERNEL32.DLL] ⇒ [ADVAPI32.DLL]
[VERSION.DLL, KERNEL32.DLL] ⇒ [SHELL32.DLL]
[OLE32.DLL, KERNEL32.DLL] ⇒[USER32.DLL]
[MPR.DLL, VERSION.DLL, KERNEL32.DLL] ⇒ [ADVAPI32.DLL]
[SHLWAPI.DLL, OLE32.DLL, ADVAPI32.DLL, KERNEL32.DLL, USER32.DLL] ⇒ [SHELL32.DLL]
[MPR.DLL, ADVAPI32.DLL] ⇒ [VERSION.DLL]
[GDI32.DLL, ADVAPI32.DLL, KERNEL32.DLL, USER32.DLL] ⇒ [OLE32.DLL]
[OLE32.DLL, SHELL32.DLL, KERNEL32.DLL, USER32.DLL] ⇒ [ADVAPI32.DLL]

Table IV shows a small portion of the association rules set we produced by applying our CRSTATIC tool. These rule sets are used as a signature to detect the ransomware family. In particular, the first row shows that if *COMCTL32.DLL, SHELL32.DLL, USER32.DLL* implies *KERNEL32.DLL*. Similarly, ninth row shows *MPR.DLL, ADVAPI32.DLL, SHELL32.DLL, USER32.DLL* implies again *KERNEL32.DLL* which also has DLL specific to crypto-ransomware as shown in Table III. The last row shows *OLE32.DLL, SHELL32.DLL, KERNEL32.DLL, USER32.DLL* implies *ADVAPI32.DLL*. If any unknown binary file matches 60 percent of these rule sets it is categorized as a ransomware with an accuracy of 70 percent.

C. Function calls specific to Crypto-Ransomware

As shown in Table III, these are the functions used by crypto-ransomware samples. As shown in Figure 1, crypto-ransomware components include: *encryption key generation*, *encrypt files*, *destroy key*. The CRSTATIC tool builds the signature database using these function calls.

VII. CONCLUSION

The traditional method of generating signatures based on the hash function of a malware is not effective for ransomware since it can be easily bypassed. In this work, we proposed CRSTATIC, a ransomware static analyzer which focused on building signatures using reverse engineering, similarity score and data mining approach based on the FP-Growth algorithm. The signature is generated statically without executing the malware sample. Experimental evaluation and results show that CRSTATIC is able to detect ransomware attacks without major performance overhead. Our contributions can be summarized as follows:

- We designed CRSTATIC, an automatic static analysis tool that leveraged the assembly instructions, PE file format, and function calls to enhance signature for anti-virus engine.
- CRSTATIC can detect the crypto-ransomware with semantically similar function blocks.
- We built the signature database specific to crypto-ransomware using the result of CRSTATIC.

REFERENCES

[1] Indicators associated with wannacry ransomware. https://www.us-cert.gov/ncas/alerts/TA17-132A. (2017).

[2] Internet security threat report. https://www.symantec.com/content/dam/symantec/docs/reports/istr-21-2016-en.pdf. (2016).

[3] pe-parse. https://github.com/trailofbits/pe-parse.

[4] Samsam - the evolution continues netting over $325,000 in 4 weeks. https://blog.talosintelligence.com/2018/01/samsam-evolution-continues-netting-over.html.

[5] thezoo - project created to make the possibility of malware analysis open. https://github.com/ytisf/theZoo.

[6] M. M. Ahmadian, H. R. Shahriari, and S. M. Ghaffarian. Connection-monitor & connection-breaker: A novel approach for prevention and detection of high survivable ransomwares. In *Information Security and Cryptology (ISCISC), 2015 12th International Iranian Society of Cryptology Conference on*, pages 79–84. IEEE, 2015.

[7] D. Bilar et al. Statistical structures: Fingerprinting malware for classification and analysis. *Proceedings of Black Hat Federal 2006*, 2006.

[8] K. Cabaj, M. Gregorczyk, and W. Mazurczyk. Software-defined networking-based crypto ransomware detection using http traffic characteristics. *arXiv preprint arXiv:1611.08294*, 2016.

[9] K. Cabaj and W. Mazurczyk. Using software-defined networking for ransomware mitigation: the case of cryptowall. *IEEE Network*, 30(6):14–20, 2016.

[10] R. Chong. Locky ransomware distributed via docm attachments in latest email campaigns. *FireEye (17 Aug 2016) Accessed Sep*, 2016.

[11] L. Constantin. Cryptowall ransomware held over 600k computers hostage, encrypted 5 billion files. *IDG News Service*, 29, 2014.

[12] A. Continella, A. Guagnelli, G. Zingaro, G. De Pasquale, A. Barenghi, S. Zanero, and F. Maggi. Shieldfs: a self-healing, ransomware-aware filesystem. In *Proceedings of the 32nd Annual Conference on Computer Security Applications*, pages 336–347. ACM, 2016.

[13] E. Eilam. *Reversing: secrets of reverse engineering*. John Wiley & Sons, 2011.

[14] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. *SIGMOD Rec.*, 29(2):1–12, May 2000.

[15] F. L. Janis, J. D. Aman, and D. R. Cox. Dynamic link libraries system and method, Sept. 21 1993. US Patent 5,247,681.

[16] K. Jarvis. Cryptolocker ransomware, 2013. *URL http://www. secureworks. com/cyber-threat-intelligence/threats/cryptolocker-ransomware/. R etrieved on April*, 21:30–31, 2014.

[17] F. Mercaldo, V. Nardone, and A. Santone. Ransomware inside out. In *Availability, Reliability and Security (ARES), 2016 11th International Conference on*, pages 628–637. IEEE, 2016.

[18] C. Moore. Detecting ransomware with honeypot techniques. In *Cybersecurity and Cyberforensics Conference (CCC), 2016*, pages 77–81. IEEE, 2016.

[19] M. Pietrek. Inside windows-an in-depth look into the win32 portable executable file format. *MSDN magazine*, 17(2), 2002.

[20] W. Roche. The advanced encryption standard, the process, its strengths and weaknesses. *University of Colorado, Denver, Spring*, 2006.

[21] M. H. U. Salvi and M. R. V. Kerkar. Ransomware: A cyber extortion. *Asian Journal of Convergence in Technology*, 2016.

[22] M. Santosa. Understanding elf using readelf and objdump. *Linux Forms article*, pages 1–6, 2006.

[23] N. Scaife, H. Carter, P. Traynor, and K. R. Butler. Cryptolock (and drop it): stopping ransomware attacks on user data. In *Distributed Computing Systems (ICDCS), 2016 IEEE 36th International Conference on*, pages 303–312. IEEE, 2016.

[24] M. Stevanovic. Linux toolbox. In *Advanced C and C++ Compiling*, pages 243–276. Springer, 2014.

[25] K. P. Subedi, D. R. Budhathoki, B. Chen, and D. Dasgupta. Rds3: Ransomware defense strategy by using stealthily spare space. In *Computational Intelligence (SSCI), 2017 IEEE Symposium Series on*, pages 1–8. IEEE, 2017.

[26] M. Sutton, A. Greene, and P. Amini. *Fuzzing: brute force vulnerability discovery*. Pearson Education, 2007.

[27] V. Total. Virustotal-free online virus, malware and url scanner. *Online: https://www. virustotal. com/en*, 2012.

[28] J. H. Yang and Y. Ryu. Design and development of a command-line tool for portable executable file analysis and malware detection in iot devices. *International Journal of Security and Its Applications*, 9(8):127–136, 2015.