

A Framework for Supporting Ransomware Detection and Prevention Based on Hybrid Analysis

Alfredo Cuzzocrea[1(✉)], Francesco Mercaldo[2], and Fabio Martinelli[2]

[1] University of Calabria, Rende, Italy
alfredo.cuzzocrea@unical.it
[2] IIT-CNR, Pisa, Italy
{francesco.mercaldo,fabio.martinelli}@iit.cnr.it

Abstract. Ransomware is a very effective form of malware, which recently raised a lot of attention since an impressive number of workstations was affected. This malware is able to encrypt the files located in the infected machine and block the access to them. The attackers will restore the machine and files only after the payment of a certain amount of money, usually given in bitcoins. In this paper we discuss an hybrid framework, combining static and dynamic analysis, exploiting APIs to prevent and mitigate ransomware threats. The evaluation, considering 1000 legitimate and ransomware applications, demonstrates that the hybrid API calls-based detection can be proved to be a promising direction in ransomware prevention and mitigation.

Keywords: Ransomware · Malware · Static analysis · Dynamic analysis · Hybrid analysis · Security

1 Introduction

Ransomware is a malware able to encrypts documents, essentially locking users out of their computers. Attackers in order to give back the original data try to extort money to restore access.

The main difference between ransomware and the other widespread malware families is represented by the exhibited behaviour. As a matter of fact, as demonstrated in [30], generic malware focuses on gathering and sending to the attackers user sensitive and private information. Moreover, malware is usually coded to be silent in order to continuously perpetuate its harmful actions. From the other side, ransomware is a threat in the malware landscape that aims at making users aware of its presence by locking the screen and, in most dangerous cases, ciphering data on the infected machine.

During the past few years, ransomware has become a relevant and emerging security problem, due to the rapid spread and the devastating consequences which can bring. Law enforcement agencies from all around Europe teamed up with antimalware vendors and other IT security companies to form the *No More*

© Springer Nature Switzerland AG 2021
O. Gervasi et al. (Eds.): ICCSA 2021, LNCS 12951, pp. 16–27, 2021.
https://doi.org/10.1007/978-3-030-86970-0_2

Ransom! project[1]. According to McAfee security analysts [28], 2016 might be remembered as "the year of ransomware", confirming the predictions of exponential growth of these kinds of attacks [25]. As an example, the WannaCry ransomware attack in May 2017 rendered unusable over 200,000 computers in more than 150 countries with ransom demands[2]. At the end of 2017, the first ransomware as a service frameworks (the so-called RaaS) appeared and this trend, as pointed out by Sophos experts, is growing up during 2018. RaaS allows attackers to rent ransomware infrastructure rather than develop it themselves[3]. The RaaS paradigm works in the following way: the attacker generally pays an upfront fee, and the author of the RaaS keeps a small percentage of each ransom paid. Generally, the rentee is allowed to set the ransom price and build the attack campaign.

From the defensive side, a weak point of the current antimalware solutions is their main operating mechanism: they mostly apply a signature-based detection; this approach requires the vendor's awareness of the malware, this is the reason why a threat must be widespread in order to be recognized.

For this reason current anti-malware software fails in identifying several ransomware threats, and could play no role in the last set of attacks (WannaCry, Petya, etc.), since new ransomware samples, whose binary signature is unknown, are continuously produced and spread in the wild as zero-day attacks. As a matter of fact, zero-day attacks are able to elude commercial anti-malware, requiring thus more advanced techniques to be detected. For instance, the Satan service claims to generate working ransomware samples and allows to set your own price and payment conditions, collects the ransom on the customer behalf and finally provides a decryption tool to victims who pay the ransom: its creators keep the remaining 30% of income, so if the victim pays a ransom worth 1 bitcoin, the customer receives 0.7 in bitcoin.

Considering that ransomware damages cost $5 billion in 2017 and $11.5 billion in 2019 (those figures are up from just $325 million in 2015[4]) ransomware detection represents a challenge also for scientific community.

Regarding the current state-of-the-art related to ransomware analysis, authors in [40] presented EldeRan, a model which dynamically monitors the initial activities of a sample, checking for specific characteristics and actions in order to classify this sample as ransomware or as benign. Unveil [27] is also a system, which dynamically tries to identify ransomware: the main functionality of this model is to monitor system's desktop and look for any changes which indicate the presence of a ransomware. In [6] authors present a SDN based analysis for detecting malware which utilizes asymmetric cryptography, i.e. CryptoWall or Locky. In their study they observe the network communication and the HTTP messages of the under examination samples and are able to detect malicious behavior. CryptoDrop is presented in [39] and it is a system which

[1] https://www.nomoreransom.org.
[2] http://wapo.st/2pKyXum?tid=ss_tw&utm_term=.6887a06778fa.
[3] goo.gl/N7PAjh.
[4] goo.gl/Gyrt1N.

monitors the activity of an application during its execution, aiming at detecting changes in a big amount of files within a system.

Behavioral analysis is the key in detecting and preventing the action of ransomware. In fact, even if they show different code, the ransomware behavior is very similar for all the samples falling in this category. Unfortunately, these common characteristics can be detected only at run-time. In this work we propose an hybrid static/dynamic framework for malware detection and effect mitigation, which exploit static analysis to identify features that can be monitored at run-time and exploited to prevent the malicious action. In the following we will introduce the framework design, the methodology and promising experimental results. Starting from the analysis of current literature, it emerges that there is no work aiming to prevent and mitigate ransomware behaviours (as a matter of fact, all the cited work are focused on ransomware detection), this is the reason why in this paper we design a framework with the aim to prevent and mitigate ransomware threats. A high-level design and a preliminary experiment about the effectiveness of the proposed framework are provided.

As regards future conceptual extensions, the topics we investigate in this paper has relevant correlations with the emerging big data trend (e.g., [17–21]). This synergy turns to be a critical aspect to be considered by future research efforts.

The paper proceeds as follows: in the next section the proposed methodology is described, in Sect. 3 the results of the experimental analysis is presented, Sect. 4 discusses the current state-of-the-literature in ransomware detection and, finally, conclusion and future work are drawn in the last section.

2 Methodology

This section presents the architecture of the proposed framework for ransomware mitigation and prevention. Two scenarios are being considered, one related to the in vitro analysis (depicted in Fig. 1), and the other one related to the in vivo analysis (depicted in Fig. 1).

Scenario I in Fig. 1 represents the workflow of the proposed framework aimed to ransomware identification.

As anticipated, a hybrid static/dynamic analysis is exploited in order to obtain information concerning the APIs invoked by the analyzed ransomware samples. More specifically:

– Through static analysis, we obtain from the executable under analysis a list of the APIs and libraries being invoked, using a reverse engineering process. Whether the gathered information match the ones stored into the SA knowledge repository (containing information related to APIs and libraries of widespread ransomware samples), the analyzed sample will be marked as “ransomware”, otherwise it will be marked as “suspicious” and will be sent to the dynamic analysis module.

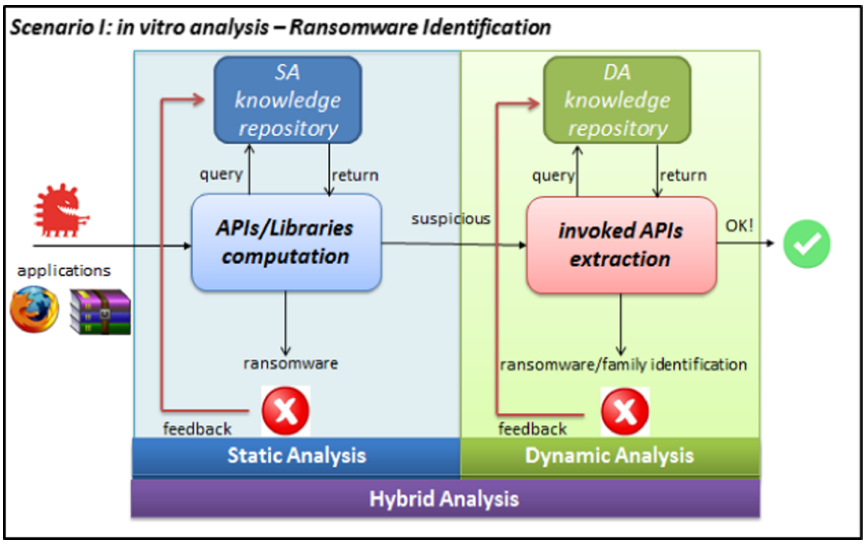

Fig. 1. Scenario I: identification of ransomware samples with static and dynamic analysis with knowledge-based feedbacks.

– From dynamic analysis, we obtain the frequency of the APIs invoked at runtime. As a matter of fact, techniques like dynamic loading (i.e., a mechanism by which a computer program is able at run time to load, execute and unload a library into memory) or reflection (i.e., the ability of a computer program to examine, introspect, and modify its own structure and behavior at runtime) can easily elude the static analysis. In addition, several obfuscation techniques are able to cipher the name of the invoked API functions, aiming by this way to evade the current signature-based antimalware mechanism. Whether the invoked APIs gathered from this module match the ones stored in the DA knowledge repository (containing information related to the APIs invoked by ransomware samples) the executable under analysis will be marked as "ransomware", otherwise, the evaluated application will be marked as "legitimate". Furthermore, in this step we plan to assign the sample in a specific malware family, if it is characterized as ransomware.

Moreover, the proposed framework provides a feedback system: once a ransomware is identified, the gathered features (from both static and dynamic analysis) are stored in the correspondent knowledge repository, in order to increase the performance of the identification and to reduce the response time of the framework (the feedback system is highlighted with red arrows in Fig. 1).

Figure 2 represents the ransomware mitigation and prevention process.

The Scenario II (the in-vivo analysis) starts when the user performs operations related to email attachment/file opening or when he/she browses an URL. A daemon installed on the user computer is in charge to collect APIs in real-time and to send the trace to the in vitro analysis for deep analysis. Whether the

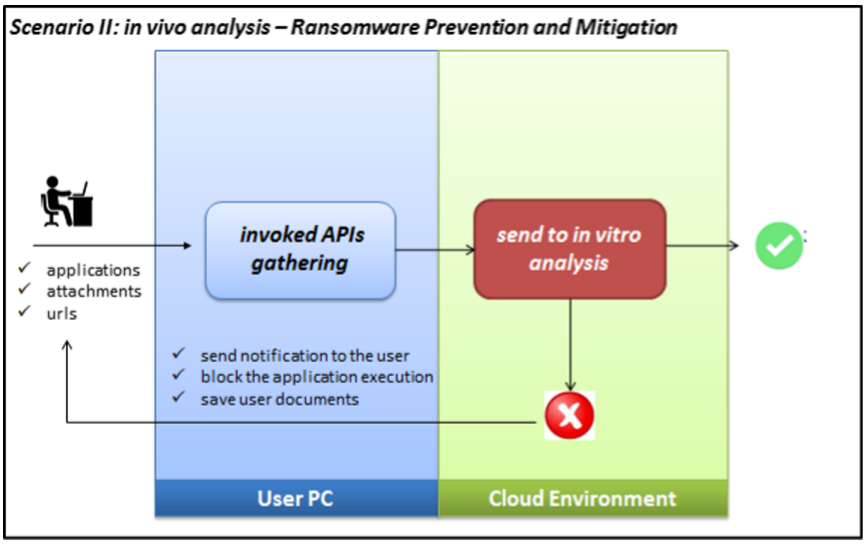

Fig. 2. Scenario II: ransomware prevention and mitigation

trace is labeled as ransomware, the deamon on the user computer performs tree different operations: (i) send notification about the maliciousness of the application to the user, (ii) block the application execution (prevention) and (iii) save user documents that could be encrypted in case the daemon is not able to kill the ransomware main process (mitigation).

3 Experimental Analysis

In this section a set of results is presented, aiming to prove the discrimination between ransomware and benign applications based on the APIs calls (provided by the static analysis) and the frequency of the invocations (provided by the dynamic analysis), as depicted in Fig. 1.

To perform the experimental evaluation we exploited the Cuckoo Sandbox [36], a widely used tool for malware analysis, which has the ability to analyze different malicious files and report back log files regarding a huge amount of information such as API calls, network traffic, behavioral and memory analysis. In this study Cuckoo is considered because of its widely use with regard to malware inspection. A number of studies and research works have also exploited this tool as for instance, in [34] the authors extend its functionality in order to execute malware samples multiple times, under different circumstances and study their overall behavior. In [35] the authors use Cuckoo in order to extract network indicators of compromise from malware samples. Another study is [11], where the authors used Cuckoo to analyze malware and dynamically reconfigure a SDN network, based on the behavior of the sample.

As depicted in Fig. 1 we compute libraries and APIs called by the applications under analysis (with regard to the static analysis) and the frequency of the API calls (with regard to the dynamic analysis). We considered APIs aimed to ciphering operations and network communication, since ransomware need usually to communicate with a Command & Control server, either to save the encryption key or use it for the ransom payment.

For this study, the libraries and the APIs taken into consideration are related mostly to ciphering operations and also to network communication, since usually a ransomware need to communicate with a Command & Control server, either to save the encryption key or use it for the ransom payment. We evaluated 500 real-world samples belonging to WannaCry family and 500 well-known legitimate applications (250 of them with ciphering abilities and the remaining 250 with no ciphering abilities). To ensure their trustworthiness, component that creates and manipulates the standard elements of the Windows user interface, such as the desktop, windows, and menus. It thus enables programs to implement a graphical user interface (GUI) Results are shown in Tables 1, 2, 3. More specifically, Table 1 presents the libraries being invoked from the WannaCry samples and the percentage of their invocation both from the ransomware samples and the benign applications. The outcome of this analysis is that in most of the ransomware samples there are calls to specific APIs, which are not used neither from application with no ciphering functionality, nor from applications with encryption capabilities i.e., *ws2_32.dll* (the Windows Sockets API used by most Internet and network applications to handle network connections), *iphlpapi.dll* (a dynamic link library) and *wininet.dll* (related to Internet functions used by Windows applications) libraries: this is symptomatic of the massive use of the network for the C&C communication usually performed by ransomware samples. Table 2, shows a list of the most common APIs used for encryption and network connection and the percentage of their presence in the under analysis samples. The interesting fact which can be observed in this Table is that the majority of the Wannacry samples uses the *LdrGetProcedureAddress* API which includes a number of encryption functions (i.e. CryptGenKey, CryptAcquireContextW, CryptAcquireContextW etc.) and the *LdrLoadDll* API, which gives the possibility to an application to drop its own .dll files to the system. Moreover, it can also been observed that the ransomware samples make use of APIs related to the connection with the internet, a behavior that is not presented by the benign applications. Finally, Table 3, presents the average frequency of the API calls, for WannaCry samples (W column in Table 3), Trusted Ciphering Applications (TC column in Table 3) and two Trusted Non Ciphering Applications (TNC column in Table 3). A first conclusion which can be extracted is that ransomware samples make in total less API calls, which is reasonable since their main functionality is the encryption of the files. In addition, regarding the two aforementioned APIs, we can also observe that they are being called more times in the WannaCry samples, rather than in the benign applications: this is symptomatic that ransomware performs its harmful behaviour continuously (as a matter of fact the encryption functions is called for each file to cipher).

Table 1. Invoked libraries

	Wannacry samples	Trusted ciphering apps	Trusted non ciphering apps
kernel.dll	100%	100%	90%
user32.dll	20%	80%	90%
advapi32.dll	90%	80%	70%
msvcrt.dll	90%	20%	10%
ws2_32.dll	80%	0%	0%
iphlpapi.dll	80%	0%	0%
wininet.dll	80%	0%	0%

Table 2. Invoked APIs

	Wannacry samples	Trusted ciphering apps	Trusted non ciphering apps
LdrGetProcedureAddress	90%	80%	40%
EncryptMessage	80%	20%	30%
DecryptMessage	10%	20%	30%
CryptExportKey	10%	0%	0%
CryptEncrypt	10%	0%	0%
CryptDecrypt	10%	0%	0%
LdrLoadDll	90%	80%	0%
InternetOpenA	80%	0%	0%
WSAStartup	80%	0%	10%
InternetOpenUrlA	80%	0%	0%
InternetCloseHandle	80%	0%	10%

Table 3. Frequency of API calls

	W	TC	TNC
Total API calls	893	37.812	125.284
LdrGetProcedureAddress	19.8%	4.57%	2.99%
LdrLoadDll	3.79%	0.32%	0.19%

The obtained results of this investigation demonstrate that the usage of API calls and invocations to discriminate between ransomware and legitimate applications seem to be a promising viable way for ransomware identification.

4 Related Work

In this section we present a review of the current literature regarding the area of ransomware detection, mainly based on the detection of malicious samples (i.e., malware detection) but also related to family identification and phylogenetic analysis.

Several studies try to propose solutions for the identification of a malware's family. More specifically, the authors of [33] present a solution which is based on the BIRCH clustering algorithm. After extracting static and dynamic analysis features from the malware samples, they construct the necessary vectors for the clustering algorithm and group the samples into families based on these information. Another study which uses the call graph clustering, is proposed in [26]. The authors base their analysis on graph similarities through the minimization of the graph edit distance. With respect to both of these studies, the proposed model of this paper uses the phylogenetic analysis and is targeting on ransomware family identification.

Another approach is that of [29], where the authors extract n-gram features from the binary code of the samples. These features were selected using the Sequential Floating Forward Selection (SFFS) method with three classifiers and the accuracy of the method reached the 96.64%. However, their methodology is different from the one presented in this paper and they do not provide results specifically referred to ransomware.

In [2] the authors make use of the PrefixSpan algorithm in order to create groups of malware families, based on sequence patterns. Although, these patterns are related only to network traffic packets and not to the overall behavior of the malware. In [43] ARIGUMA is proposed, a malware analysis system for family classification. The authors consider the number of functions being called in a function and the number of functions which call a specific function. Also, the local variables, the arguments and the instructions are taken into account. Even though, this method can also detect obfuscated APIs, the classification accuracy is only 61.6%. In [24] the authors present a malware family classification system which is based on the instruction sequence. The analysis is made through three steps: the feature extraction, the classification and the signature generator. The accuracy of their method reaches up to 79%, which is less than the one achieved in this paper.

Phylogenetic analysis has been used in a number of studies regarding malware. More specifically, in [8] the authors use the phylogenetic analysis in order to assign android malware samples in families. They consider the sequence of the system calls and based on these features they compute the similarity functions. The main difference between this work and the proposed one, is that the second focuses on the ransomware and is also able to distinguish trusted applications. Another study is that of [9], where the authors exploit phylogenesis so as to compute the relationship between 6 groups of malware. In [41] a method of extracting API calls out of malware samples and calculating the similarity amongst them, is presented. Although, this study is focused on analyzing the malware behavior and not on classifying the samples into families. A study which

also uses phylogenetic trees is [42], where the authors analyze internet worms and network traffic and present families of worms can be related by their normalized compression distance. Martinelli and colleagues [30] adopt supervised classification algorithms for ransomware family identification. Moreover they consider the binary trees generated by these algorithms to infer the phylogenetic relationships. They experiment the proposed method on real-world ransomware applications belonging to three widespread families (i.e., petya, badrabbit and wannacry),

A phylogenetic analysis related to Android malware is proposed authors in [12,14] where authors consider model checking techniques for malware evolution tracking and for discovering ancestor and descendant relationship.

5 Conclusion and Future Work

Considering the massive damage recently caused by ransomware, the development of techniques and methodologies focused to the identification and mitigation are needed. In this paper we proposed a hybrid framework that exploiting API calls (by static analysis) and invocations (by dynamic analysis) is designed to prevent the ransomware threat. We evaluated, using the Cuckoo framework, the effectiveness of API calls and invocations to discriminate between ransomware and legitimate applications, obtaining encouraging results. As future work we plan to evaluate the proposed framework and evaluate their effectiveness on a larger set of applications, both benign and malicious. In addition we plan to consider the adoption of formal methods [1,37,38] for reaching better accuracy for the ransomware detection and mitigation tasks, considering that formal methods already showed their efficacy in other fields as, for instance, as biology [10,23], medicine [3–5] but also software security [7,13,15,16,22,31,32].

Acknowledgments. This work has been partially supported by MIUR - SecureOpenNets, EU SPARTA, CyberSANE and E-CORRIDOR projects.

References

1. Barbuti, R., De Francesco, N., Santone, A., Vaglini, G.: Reduced models for efficient ccs verification. Formal Methods Syst. Des. **26**(3), 319–350 (2005)
2. Boukhtouta, A., Lakhdari, N.E., Debbabi, M.: Inferring malware family through application protocol sequences signature. In: 2014 6th International Conference on New Technologies, Mobility and Security (NTMS), pp. 1–5, March 2014
3. Brunese, L., Mercaldo, F., Reginelli, A., Santone, A.: Formal methods for prostate cancer gleason score and treatment prediction using radiomic biomarkers. Magn. Reson. Imaging **66**, 165–175 (2020)
4. Brunese, L., Mercaldo, F., Reginelli, A., Santone, A.: Neural networks for lung cancer detection through radiomic features. In: 2019 International Joint Conference on Neural Networks (IJCNN), pp. 1–10. IEEE (2019)

5. Brunese, L., Mercaldo, F., Reginelli, A., Santone, A.: An ensemble learning approach for brain cancer detection exploiting radiomic features. Comput. Methods Programs Biomed. **185**, 105134 (2020)
6. Cabaj, K., Gregorczyk, M., Mazurczyk, W.: Software-defined networking-based crypto ransomware detection using HTTP traffic characteristics. Comput. Electr. Eng. **66**, 353–368 (2018)
7. Canfora, G., Mercaldo, F., Moriano, G., Visaggio, C.A.: Composition-malware: building android malware at run time. In: 2015 10th International Conference on Availability, Reliability and Security (ARES), pp. 318–326. IEEE (2015)
8. Canfora, G., Mercaldo, F., Pirozzi, A., Visaggio, C.A.: How i met your mother? In: Proceedings of the 13th International Joint Conference on e-Business and Telecommunications, pp. 310–317. SCITEPRESS-Science and Technology Publications, Lda (2016)
9. Carrera, E., Erdélyi, G.: Digital genome mapping - advanced binary malware analysis (2004)
10. Ceccarelli, M., Cerulo, L., Santone, A.: De novo reconstruction of gene regulatory networks from time series data, an approach based on formal methods. Methods **69**(3), 298–305 (2014)
11. Ceron, J.M., Margi, C.B., Granville, L.Z.: MARS: an SDN-based malware analysis solution. In: 2016 IEEE Symposium on Computers and Communication (ISCC), pp. 525–530, June 2016
12. Cimino, M.G.C.A., De Francesco, N., Mercaldo, F., Santone, A., Vaglini, G.: Model checking for malicious family detection and phylogenetic analysis in mobile environment. Comput. Secur. **90**, 101691 (2020)
13. Cimitile, A., Martinelli, F., Mercaldo, F., Nardone, V., Santone, A.: Formal methods meet mobile code obfuscation identification of code reordering technique. In: 2017 IEEE 26th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), pp. 263–268. IEEE (2017)
14. Cimitile, A., Martinelli, F., Mercaldo, F., Nardone, V., Santone, A., Vaglini, G.: Model checking for mobile android malware evolution. In: 2017 IEEE/ACM 5th International FME Workshop on Formal Methods in Software Engineering (FormaliSE), pp. 24–30. IEEE (2017)
15. Cimitile, A., Mercaldo, F., Nardone, V., Santone, A., Visaggio, C.A.: Talos: no more ransomware victims with formal methods. Int. J. Inf. Secur. **17**(6), 719–738 (2018)
16. Ciobanu, M.G., Fasano, F., Martinelli, F., Mercaldo, F., Santone, A.: Model checking for data anomaly detection. Procedia Comput. Sci. **159**, 1277–1286 (2019)
17. Cuzzocrea, A.: Improving range-sum query evaluation on data cubes via polynomial approximation. Data Knowl. Eng. **56**(2), 85–121 (2006)
18. Cuzzocrea, A., Matrangolo, U.: Analytical synopses for approximate query answering in OLAP environments. In: Galindo, F., Takizawa, M., Traunmüller, R. (eds.) DEXA 2004. LNCS, vol. 3180, pp. 359–370. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30075-5_35
19. Cuzzocrea, A., Moussa, R., Xu, G.: OLAP*: effectively and efficiently supporting parallel OLAP over big data. In: Cuzzocrea, A., Maabout, S. (eds.) MEDI 2013. LNCS, vol. 8216, pp. 38–49. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-41366-7_4
20. Cuzzocrea, A., Saccà, D., Serafino, P.: A hierarchy-driven compression technique for advanced OLAP visualization of multidimensional data cubes. In: Tjoa, A.M., Trujillo, J. (eds.) DaWaK 2006. LNCS, vol. 4081, pp. 106–119. Springer, Heidelberg (2006). https://doi.org/10.1007/11823728_11

21. Cuzzocrea, A., Serafino, P.: LCS-hist: taming massive high-dimensional data cube compression. In: Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology, pp. 768–779 (2009)
22. Martinelli, F., Mercaldo, F., Orlando, A., Nardone, V., Santone, A., Sangaiah, A.K.: Human behaviour characterization for driving style recognition in vehicle system (2018)
23. Francesco, N.D., Lettieri, G., Santone, A., Vaglini, G.: Grease: a tool for efficient "nonequivalence" checking. ACM Trans. Softw. Eng. Methodol. (TOSEM) **23**(3), 24 (2014)
24. Huang, K., Ye, Y., Jiang, Q.: ISMCS: an intelligent instruction sequence based malware categorization system. In: 2009 3rd International Conference on Anti-counterfeiting, Security, and Identification in Communication, pp. 509–512, August 2009
25. Infosec Institute: Evolution in the World of Cyber Crime. Technical report, Infosec Institute, June 2016
26. Kinable, J., Kostakis, O.: Malware classification based on call graph clustering. J. Comput. Virol. **7**(4), 233–245 (2011)
27. Kirda, E.: Unveil: a large-scale, automated approach to detecting ransomware (keynote). In: 2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER), p. 1, February 2017
28. McAfee Labs: McAfee Labs Threats Report - December 2016. Technical report, McAfee Labs, August 2016
29. Liangboonprakong, C., Sornil, O.: Classification of malware families based on n-grams sequential pattern features. In: 2013 IEEE 8th Conference on Industrial Electronics and Applications (ICIEA), pp. 777–782, June 2013
30. Martinelli, F., Mercaldo, F., Michailidou, C., Saracino, A.: Phylogenetic analysis for ransomware detection and classification into families. In: ICETE, no. 2, pp. 732–737 (2018)
31. Martinelli, F., Mercaldo, F., Nardone, V., Santone, A.: Car hacking identification through fuzzy logic algorithms. In: 2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), pp. 1–7. IEEE (2017)
32. Mercaldo, F., Nardone, V., Santone, A., Visaggio, C.A.: Hey malware, i can find you! In: 2016 IEEE 25th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), pp. 261–262. IEEE (2016)
33. Pitolli, G., Aniello, L., Laurenza, G., Querzoni, L., Baldoni, R.: Malware family identification with birch clustering. In: 2017 International Carnahan Conference on Security Technology (ICCST), pp. 1–6, October 2017
34. Provataki, A., Katos, V.: Differential malware forensics. Digit. Investig. **10**(4), 311–322 (2013)
35. Rudman, L., Irwin, B.: Dridex: analysis of the traffic and automatic generation of IOCs. In: 2016 Information Security for South Africa (ISSA), pp. 77–84, August 2016
36. Cuckoo Sandbox. Cuckoo Sandbox - Automated Malware Analysis (2018). https://cuckoosandbox.org/. Accessed 06 Mar 2018
37. Santone, A.: Automatic verification of concurrent systems using a formula-based compositional approach. Acta Inf. **38**(8), 531–564 (2002)
38. Santone, A.: Clone detection through process algebras and java bytecode. In: IWSC, pp. 73–74. Citeseer (2011)
39. Scaife, N., Carter, H., Traynor, P., Butler, K.R.: Cryptolock (and drop it): stopping ransomware attacks on user data. In: 2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS), pp. 303–312, June 2016

40. Sgandurra, D., Muñoz-González, L., Mohsen, R., Lupu, E.C.: Automated dynamic analysis of ransomware: Benefits, limitations and use for detection. arXiv preprint arXiv:1609.03020 (2016)
41. Wagener, G., State, R., Dulaunoy, A.: Malware behaviour analysis. J. Comput. Virol. **4**(4), 279–287 (2008)
42. Wehner, S.: Analyzing worms and network traffic using compression. J. Comput. Secur. **15**(3), 303–320 (2007)
43. Zhong, Y., Yamaki, H., Yamaguchi, Y., Takakura, H.: Ariguma code analyzer: efficient variant detection by identifying common instruction sequences in malware families. In: 2013 IEEE 37th Annual Computer Software and Applications Conference, pp. 11–20, July 2013