Malware Attacks: A Survey on Mitigation Measures

Anna V. James and S. Sabitha

1 Introduction

Malicious software or Malware can be defined as a software that "deliberately fulfills the harmful intent of an attacker." Terms, such as "worm," "virus," "Trojan horse," "ransomware," etc., are the different classes of malware. Highlighting security vulnerabilities in the software or showoff of technical ability was the motivation for malware creators at the early times. Today, there is a flourishing underground economy based on malware. Now, it is no longer the fun factor, but the perspective of the money that can be made drives the development of such malware.

Introduction of new malware every day is a challenge to antivirus vendors. The main challenge of antivirus writers is the growing stream of obfuscated malware samples with several variations to avoid existing detection methodology. Very recent type of malware named as "ransomware" that extorts money from the victims became most popular with the cybercriminals. Compared to traditional malware, the attack pattern of ransomware is different. Traditional malware uses sophisticated coding techniques to steal the credential or to conduct any targeted attack. On the other hand, ransomware is designed purposefully to ask money from the victims by making the computer system unusable. In most of the cases, ransomware uses cryptographic technology to encrypt the user data.

It is critical to identify these types of malware, due to the fact that they cause lots of damage to large surface area. For detection, dynamic analysis is more popularly used so as to overcome the limitations of static analysis. Nowadays, Machine Learning techniques are also applied along with dynamic analysis [1, 2].

A. V. James (✉) · S. Sabitha
College of Engineering Trivandrum, Thiruvananthapuram, India
e-mail: annavjames@cet.ac.in; sabitha@cet.ac.in

M. Palesi et al. (eds.), *Second International Conference on Networks and Advances in Computational Technologies*, Transactions on Computational Science and Computational Intelligence, https://doi.org/10.1007/978-3-030-49500-8_1

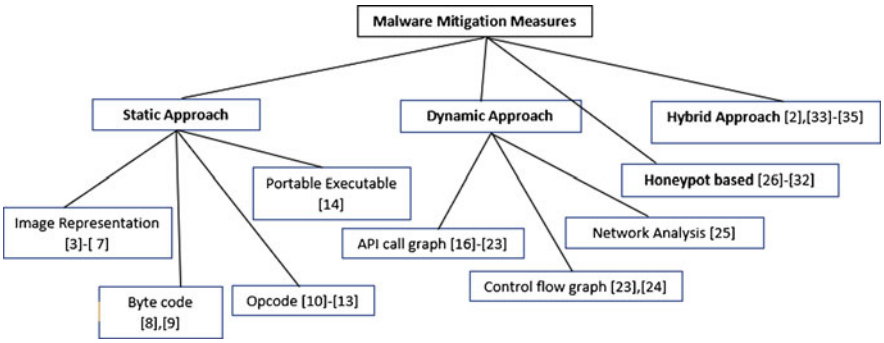

Fig. 1 Malware mitigation strategies

Antivirus (AV) vendors strive to keep the pace with sophisticated malware variants. The malware mitigation strategies used for detecting other malware variants can also be used in detection of ransomware. In this paper we tried to cover various best approaches used in detection of malwares, starting from traditional but effective signature (static) based approach to highly efficient hybrid approach. We also discuss Honeypot based approach that has been successful in detecting specific kinds of malwares like ransomwares. Figure 1 covers different techniques used in each approach. In fact, when new variants are spread, signature based mechanisms are easily deceived. Furthermore, very sophisticated packing techniques implemented by current ransomwares to evade detection, e.g. obfuscated API calls, delivering the static analysis useless.

Malware detection methods are fundamentally categorized into different categories from different points of view. They can be classified into four types as in Fig. 1: Signature based approach, Dynamic approach, Honeypot based approach, and Hybrid approach. The following sections discuss in detail about each of them.

The rest of the paper is organized as follows: In Sect. 2 we cover malware detection methods; Sect. 3 illustrates a comparison table; and finally in Sect. 4 the summary of the survey is discussed.

2 Mitigation Measures

2.1 Static Approach (Signature Based)

One of the traditional malware detection strategy is static approach. It examines the malware binaries without executing them. It is one of the fast and safe technique for malware detection. It mainly uses the hash signature, embedded strings, byte code distribution, etc. present in the malicious binary. This technique does not work well for sophisticated malware. Signatures are Short strings of bytes that are unique

for each program. However, this signature based method is not effective against modified and unknown malicious executable.

Image Representation: Binary Texture Analysis Image Representation based technique makes use of pattern based method in identification of malware signature in an image. Image texture based features can be used for various applications such as image classification, image search, etc. Image representation of the malware binary is performed and texture based features are extracted from it as done by L. Nataraj [3]. The 2D matrix of grey scale image is generated by first converting malware binary into an array of 1D 8-bit integers as shown in Fig. 2. GIST, SIFT features are extracted from the image for further processing.

In [4], L. Nataraj use a traditional approach for classification of grey scale image. They use the idea that the variants of a malware have similar images and different malware have different images. In [5] S. Choi et al. propose a deep learning based method for malware classification by using the above method. The extracted feature set from malware is used to train kNN classifier. Nataraj [4] uses GIST to compute the mean value of magnitude of local feature and generate 320-dimensional GIST feature vector. Gibert [6] uses the binary image as input to CNN.

Other approaches [7] make use of deep Convolutional models to classify image based on LBP features. In addition to that, new variants of malware are created by changing only limited part of code hence, images can be used to detect slight changes by retaining the overall structure. Convolutional neural networks are found to be the best model used for image classification problems.

Byte Code Sequence Byte level details in a malware binary can be used for finding the relation with other binary files. Use of n-grams byte features for detecting malware has been done by E. Raff et al. in [8]. They treat the malicious binary as sequence of bytes and n consecutive bytes are considered as individual feature. It looks for the unique combination of n byte grams. Different works check for $n = 1$ to $n = 8$ bytes, while Tabish et al. [9] worked on byte level file content in a block-wise manner. Finally, the block-wise classification results of a given file are correlated to classify it as benign or malware. Byte code sequence requires no knowledge of format of file and is harmless in nature.

Opcode Sequence Opcodes and byte sequences can be interchangeably used for malware detection. But, the main advantage of opcode over byte code in malware

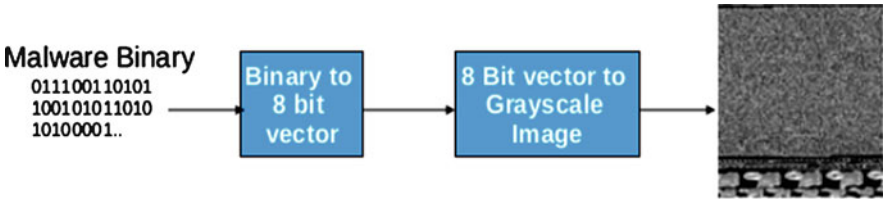

Fig. 2 Malware as 2D image

detection is that, opcode is efficient in detecting obfuscated and metamorphic malware. Malware executable and benign executable have different frequency distribution of opcodes (say mov, push). The similarity degree of two executable is compared based on these features. By comparing opcode distribution in malicious and non-malicious samples, detection and differentiation of advanced malware can be done.

Akkas et al. [10] performed assembly analysis on the ransomware samples and succeeded to figure out how the first files are created and how user's files are encrypted and also were able to identify the beginnings of the threads that encrypt the data.

Bilar [11] does the most significant research on OpCodes. His works proved that single OpCodes can be used as a feature in malware detection. For that, he analyzed the capability of single OpCodes statistically and demonstrated their reliability to determine the maliciousness of an executable. He also proved that OpCodes can be used as a powerful representation for executable files.

In [12] I. Santos et al. proposed a method which relies on the frequency of presence of opcode sequences. This approach is not effective for packed malware.

Runwal et al. [13] proposed another method that can be used for detecting unknown as well as metamorphic malware families by comparing closeness of simple graph. For that, they extracted OpCodes from malware and benign types, and occurrence frequency is noted for each pair opcode. This value is used to construct graph and is used to predict the maliciousness of a new executable by calculating the closeness of graph. Frequency distribution and graph construction from opcodes are two efficient approaches that can easily detect obfuscated malware samples.

Portable Executable Portable Executable (PE) is a file format of executable files in Windows Operating System. Most of the virus reported so far belong to PE type [14]. Viruses like CodeRed, Killonce, CIH, CodeBlue, Nimda, Sobig, Sircam, and Love Gate aim at PE files. The essential information used to load a PE file is contained in DOS MZ header and all PE files start with it. The PE file comprises of multiple sections and each section contains data with common attributes. PE parser extracts the APIs called by a PE file from the import table. For static extraction of API execution calls, PE parser extracts a 32-bit unique global API ID.

Ye et al. [14] proposed a system resting on the analysis of Windows APIs invoked by PE files, and using Objective-Oriented Association (OOA) mining developed an Intelligent Malware Detection System (IMDS) for classification.

2.2 Dynamic Approach (Behavior Based)

As static approach captures only the information available in the malware executable, it can be easily evaded using simple obfuscation techniques. Hence, Dynamic approach which analyses the malware at run time can be used to analyze the specific behavior of malware [1, 15]. Behavior based analysis can be efficiently

used to detect several families of malware by inspecting what it does rather than what it says. It involves more complex tasks in acquiring and extraction of dynamic features from malware logs. But still dynamic approach is efficient in detection. These mechanisms help in detecting the program that generates new mutants continuously.

API Call Graph API, Application Programming Interface, is used by almost all programs to send requests to operating system. One of the most attractive way that represents the malware behavior is by API calls. Hofmeyr et al. were the first to use sequences of API call for constructing feature set of malware [16]. They used system call sequences for performing anomaly based detection. Short system call sequences were used to make the behavior profile of normal behavior. Hamming distance values above user-specified threshold are reported as anomalies. Afterward, an extensive research was made on using API calls by Bergeron et al. [17], Sekar et al. [18], and Sung et al. [19], etc. Even though it performed well in malware detection, there have been two main problems such as Handling of large set of rules for constructing the classifier and Finding effective rules to classify new file samples. By using post processing techniques of associative classification the above two problems were overcome by, Ye et al. [20]. In [21], Chi-squared testing and insignificant rule pruning are applied initially, followed by database coverage by rule ranking mechanism based on the Chi-square measure and Pessimistic error estimation. The best first rule is used for final prediction. CIDCPF is in-cooperated into existing IMDS system and generates CIMDS [20]. Post processing is used for the first time in malware detection for associative classification.

Code graph called topological graph is built from malicious and benign executables by Jeong and Lee [22] from API calls. The main drawback of this approach is that the code graph tends to be too large. Hence, the size of code graph is reduced by Lee et al. [23] by classifying API calls into 128 groups.

Control Flow Graph If a graph can be drawn representing the control flow of a program, it can be used for analyzing the behavior of the program. A directed graph called Control Flow Graph is used for the same. Each node in the graph represents the statement of the program and the edge between the nodes represents control flow between the statements. The statements can be either assignment statement in the program, copy statement, branches, etc. J. Lee et al. in [23] represent the malware detection as sub-graph isomorphism problem. A set of normalization operation is performed on the executable after disassembling it. It can reduce the effect of mutation techniques and can unveil the flow connections between malicious and benign code. As shown in Fig. 3 the corresponding CFG is generated. Newly generated CFG is compared against the CFG of a normalized malware to check the presence of sub-graph which is isomorphic to CFG of the normalized one.

Bonfante et al. [24] use CFG as signature for detecting malware. Each assembler is composed of four types of instructions, namely conditional jumps (jcc), function calls (call), non-conditional jumps (jmp), and function returns (ret). Any contiguous sequence of instructions is abstracted between nodes named “inst” and “end.” So, they defined six types of node: inst, call, jmp, ret, jcc, and end. The CFG based on

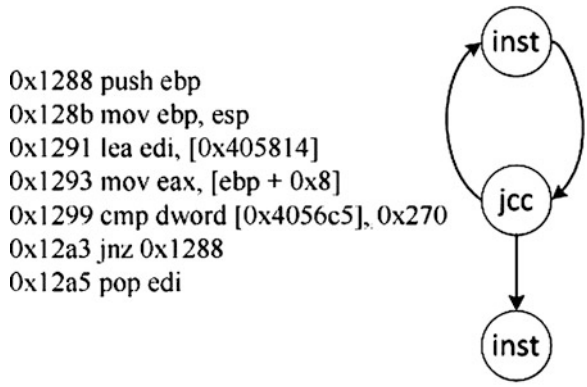

Fig. 3 CFG Extraction proposed by Bonfante et al. [24]

these types is constructed as illustrated in Fig. 3. Then, these nodes are reduced and are used as a signature for each file.

Network Analysis Dynamic malware analysis also monitors the network level activity of the malware. It tries to observe and capture the messages and packets send between the network and malware. Malware from ransomware families tends to communicate with the command and control server for its operation like encryption. This malicious traffic can be obtained using network analysis of malware. Network analysis operates over different OSI protocols like TCP/IP, HTTP, UDP, etc. Network analysis tools like Wireshark are used for this purpose. Some programs try to send and receive data requests from different IPs. These IPs would be TOR exists usually used in case of ransomwares. As specified in [25] malware analysis of ransomware includes this technique. They try to connect to several websites which are unsuccessful to connect from normal browsers. These anomalous behavior can be used to distinguish between the malware activities. Vigneswaran et al. [25] Several other intrusion detection systems try to find the anomalous behavior by using KDD data set for normal network operations and current values of these variables are used for classification.

2.3 Honeypot Based Approach

Honeyfiles are trap files employed in the deception environment in order to track and trap the malware with specific behavior.

Cryptostalker, a real time detection tool based on file system activity for windows and linux [26], when more than a specific number of files are within a time interval, it generates an advertisement. For earlier detection Pingree [27] proposes a technique. Another variant of this technique relies on the admission of the cyber kill-chain model by Hutchins et al. [28], in which an attack can occur if each step of the chain is executed sequentially.

In particular, the use of honeyfiles is an advisable mechanism in the phases of "permanence-exfiltration" and "lateral movement" in case of Ransomwares as in [29]. A real tool developed called Anti Ransom for Windows platforms [30]. In order to prevent data from ransomware and other malicious apps in Windows 10, Microsoft introduced a control folder access in [31]. R-Locker by Gomez-Hernndez et al. in [32] proposes a novel approach which in addition to detection it also thwarts the malicious activity and is specific for Unix platform. It deploys a FIFO like structure rather than normal file and can completely block the program accessing it. The cost and complexity of this solution is really low and does not impede with the normal operation of the environment. It does not require previous knowledge or training, and is efficient in fighting against unknown, zero-day attacks.

2.4 Hybrid Approach

Hybrid approach as the name suggests contains combination of several traditional approaches such as static, dynamic, etc. There are many works done on the grounds of ransomware with hybrid features in cooperating static and dynamic approaches. There have been several works done by using static and dynamic feature set. Ahmadian and Shahriari [33] in 2entFOX capture static and dynamic features of highly survivable ransomwares and designed a detection system using Bayesian belief network which uses statistical possibilities of the extracted features. The feature sets used in this system can be increased or decreased according to the security countermeasures and every module of this framework can be used in other driven systems too.

RansHunt proposed by Hasan and Rahman [34] is an Analysis Framework based on Support Vector Machines uses integrated feature set by integrating static and dynamic features. Another efficient layered approach developed by Shaukat and Ribeiro [2] is RansomWall, a Layered Defense System against Ransomware Attacks using Machine Learning. It employs a layered defense system which incorporates static analysis engine in the initial layers followed by honeyfiles and then dynamic analysis engine. It has a file backup layer to replace the modified files. It is one of the promising approaches for the early detection of ransomware.

3 Comparison

Tables 1 and 2 compare between different approaches in Static Malware analysis, Dynamic Analysis, Mitigation using Honeypot and Hybrid Approach. The comparison table is splitted as it does not fit on a single page. It also discusses the advantages and disadvantages associated with each methodology. Signature based solutions which uses opcodes, byte codes, PE headers, etc. can be used to detect malware. They are easy to implement with minimum cost and time but cannot overcome

Table 1 Comparison of different static approaches

Methodology	Refs.	Advantages	Disadvantages
Image representation	[3–7]	– Fast and Easy detection – Easily finds polymorphic code – Techniques like SIFT enable to efficiently find malware signature – Obtains large number of feature set	– Malware binary has to be converted to fixed sized images – Some portion of image would be pruned due to large size – Increases computational complexity
Byte code sequence	[8, 9]	– Easily obtained from malware executable – Feature set generation is also easy task	– Not efficient for obfuscated byte code – Large number of features
Opcode sequence	[10–13]	– Efficient than Byte code – Ability to detect obfuscated and metamorphic malwares – Different frequency distribution of opcodes enables detection easier – Reduces the false positive rate	– Takes into account only the opcode frequency – Information regarding malware behavior is not considered – It is difficult to evaluate. – Imbalance datasets
Portable executable	[14]	– PE contains implementation information of the executable – Can keep track of the API calls – Detects malware before their execution – Detects previously undetectable malicious executable – Detects borderline binaries	– Applicable only for windows executable

obfuscated malwares. In response to this, dynamic approaches were proposed, which analyze the program behavior during execution. They require a large amount of resources and have a substantial overhead on the system. Honeyfiles can be easily deployed and are cost efficient but can be applied to malware of specific types. Hybrid methodology combines the advantages of both static and dynamic approaches. Even though there are overhead on the system, the efficiency of these systems is very high.

Table 2 Comparison of dynamic, honeypot and hybrid approach

Methodology	Refs.	Advantages	Disadvantages
API call graph	[16–23]	– Detects polymorphic and unknown malware – Fewer false positives than other scanners – Outperforms other classification approaches in both detection ratio and accuracy – Detects metamorphic malware – Outperforms other classification methods in terms of performance and efficiency – Generates semantic signature	– Large set of generated rules for building classifier – Only provides binary predictions – Large size of graph for comparison
Control flow graph	[23, 24]	– Detects metamorphic malwares – High detection ratio – Low false positive rate	– Did not compare the efficiency of its algorithm with other techniques – Did not evaluate false negatives
Network analysis	[25, 25]	– Real time detection of network flow – Able to obtain specific malicious behavior – Enable to prevent malware propagation	– Cannot obtain information if highly sophisticated networks are used – Difficult in handling data of different formats
Honeypot method	[26–32]	– Simple implementation – Reduces time and space complexity – Requires no training – Detects and prevent malware	– Can only be used for specific malware types
Hybrid method	[2, 33, 34]	– Improves the accuracy of malware detection effectively – Low false positive ratio	– Increased time complexity

4 Conclusion

In this survey we discussed about several malware detection techniques employed so far especially for ransomware and proposed a unique classification scheme for malware detection techniques. The objective of the survey is to provide a procedure, which could be suitable for further studies and to develop malware detection techniques. Since there are traditional approaches that are employed still for malware detection, this survey focused on various other heuristic methods

that are successfully applied for malware detection. It also provides an insight of different detection techniques that can be employed based on the application and behavior of malware under consideration.

References

1. Homayoun, S., Dehghantanha, A., Ahmadzadeh, M., Hashemi, S., Khayami, R.: Know abnormal, find evil: frequent pattern mining for ransomware threat hunting and intelligence. IEEE Trans. Emer. Topics Comput. **8**(2), 341–351 (2017). https://doi.org/10.1109/TETC.2017.2756908
2. Shaukat, S.K., Ribeiro, V.J.: RansomWall: a layered defense system against cryptographic ransomware attacks using machine learning. In: IEEE 10th International Conference on Communication Systems Networks (2018). https://doi.org/10.1109/COMSNETS.2018.8328219
3. Nataraj, L., Yegneswaran, V., Porras, P., Zhang, J.: A comparative assessment of malware classification using binary texture analysis and dynamic analysis. In: Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence (2011). https://doi.org/10.1145/2046684.2046689
4. Nataraj, L.: Malware Images: Visualization and Automatic Classification. Vision Research Lab, University of California, Santa Barbara (2011). https://doi.org/10.1145/2016904.2016908
5. Choi, S., Jang, S., Kim, Y., Kim, J.: Malware detection using malware image and deep learning. In: 2017 International Conference on Information and Communication Technology Convergence (ICTC), pp. 1193–1195 (2017). https://doi.org/10.1109/ICTC.2017.8190895
6. Gibert, D.: Convolutional Neural Networks for Malware Classification. A thesis presented for the degree of Master in Artificial Intelligence, Universitat de Barcelona (UB) (2016)
7. Luo, J.-S., Lo, D.C.-T.: Binary malware image classification using machine learning with local binary pattern. In: IEEE International Conference on Big Data (BIGDATA) (2017). https://doi.org/10.1109/BigData.2017.8258512
8. Raff, E., Zak, R., Cox R., Sylvester, J., Yacci, P., Ward, R., Tracy, A., McLean, M., Nicholas, C.: An investigation of byte n-gram features for malware classification. J. Comput. Virol. Hacking Tech. **14**, 1–20 (2018). https://doi.org/10.1007/s11416-016-0283-1
9. Tabish, S.M., Shafiq, M.Z., Farooq, M.: Malware Detection using statistical analysis of byte-level file content. In: Conference Proceedings of the ACM SIGKDD Workshop on Cyber Security and Intelligence Informatics, Paris (2009). https://doi.org/10.1145/1599272.1599278
10. Akkas, A., Chachamis, C.N., Fetahu, L.: Malware Analysis of WanaCry Ransomware. https://courses.csail.mit.edu/6.857/2017/project/20.pdf
11. Bilar, D: Opcodes as predictor for malware. Int. J. Electron. Secur. Digit. Forensics **1**(2), 156–168 (2007). https://doi.org/10.1504/IJESDF.2007.016865
12. Santos, I., Brezo, F., Nieves, J., Penya, Y.K., Sanz, B., Laorden, C., Bringas, P.G.: Idea: opcode-sequence-based malware detection. In: International Symposium on Engineering Secure Software and Systems, pp. 35–43 (2010). https://doi.org/10.1007/978-3-642-11747-3_3
13. Runwal, N., Low, R.M., Stamp, M.: OpCode graph similarity and metamorphic detection. J. Comput. Virol. **8**(1–2), 37–52,(2012). https://doi.org/10.1007/s11416-012-0160-5
14. Ye, Y., Wang, D., Li, T., Ye, D., Jiang, Q.: An intelligent PE-malware detection system based on association mining. J. Comput. Virol. **4**(4), 323–334 (2008). https://doi.org/10.1007/s11416-008-0082-4
15. KALPA, Introduction to Malware. http://securityresearch.in/index.php/projects/malware_lab/introduction-to-malware/8/
16. Hofmeyr, S., Forrest, S., Somayaji, A.: Intrusion detection using sequences of system calls. ACM J. Comput. Secur. **6**(3), 151–180 (1998)

17. Bergeron, J., Debbabi, M., Desharnais, J., Erhioui, M.M., Tawbi, N.: Static detection of malicious code in executable programs. Int. J. Req. Eng. **2001**(184–189), 79 (2001)
18. Sekar, R., Bendre, M., Bollineni, P., Dhurjati, D.: A fast automaton-based approach for detecting anomalous program behaviors. In: Proceedings 2001 IEEE Symposium on Security and Privacy (2001). https://doi.org/10.1109/SECPRI.2001.924295
19. Sung, A.H., Xu, J., Chavez, P., Mukkamala, S.: Static analyzer of vicious executables. In: IEEE 20th Annual Computer Security Applications Conference, pp. 326–334 (2004). https://doi.org/10.1109/CSAC.2004.37
20. Ye, Y., Li, T., Jiang, Q., Wang, Y.: CIMDS: adapting postprocessing techniques of associative classification for malware detection. IEEE Trans. Syst. Man Cybern. C **40**(3), 298–307 (2010). https://doi.org/10.1109/TSMCC.2009.2037978
21. Snedecor, W., Cochran, W.: Statistical Methods, 8th edn. Iowa State University Press, Iowa City (1989)
22. Jeong, K., Lee, H.: Code graph for malware detection. In: Information Networking. ICOIN. International Conference, pp. 1–5 (2008). https://doi.org/10.1109/ICOIN.2008.4472801
23. Lee, J., Jeong, K., Lee, H.: Detecting metamorphic malwares using code graphs. In: Proceedings of the ACM Symposium on Applied Computing, pp. 1970–1977. ACM, New York (2010). https://doi.org/10.1145/1774088.1774505
24. Bonfante, G., Kaczmarek, M., Marion, J.Y.: Control Flow Graphs as Malware Signatures. WTCV (2007)
25. Vigneswaran, K.R., Vinayakumar, R., Soman, K.P., Poornachandran, P.: Evaluating shallow and deep neural networks for network intrusion detection systems in cyber security. In: Ninth International Conference on Computing, Communication and Networking Technologies (ICCCNT), Bengaluru (2018). https://doi.org/10.1109/ICCCNT.2018.8494096
26. Cryptostalker. https://github.com/unixist/randumb#cryptostalker-example
27. Pingree, L.: Emerging Technology Analysis: Deception Techniques and Technologies Create Security Technology Business Opportunities. https://www.gartner.com/doc/reprints?id=1-2LSQOX3&ct=150824&st=sb&aliId=87768
28. Hutchins, E.M., Cloppert, M.J., Amin, R.M.: Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. Lead Issues Inf Warf Secur. Res **1**, 1–14 (2011)
29. Moore, C.: Detecting ransomware with honeypot techniques. In: Cybersecurity and Cyberforensics Conference, pp. 77–81 (2016)
30. Yago, J.: Security Projects: Anti Ransom (2017). http://www.security-projects.com/?Anti_Ransom
31. GBH: Microsoft introduced a control folder access to prevent data from ransomware and other malicious apps and threats in Windows 10 insider release
32. J.A. Gmez-Hernndez, Alvarez-Gonzlez, L., Garca-Teodoro, P.: R-locker: thwarting ransomware action through a honeyfile-based approach. Comput. Secur. 73, 389–398 (2018)
33. Ahmadian, M.M., Shahriari, H.R.: 2entFOX: a framework for high survivable ransomwares detection. In: 13th International ISC Conference on Information Security and Cryptology (2016). https://doi.org/10.1109/ISCISC.2016.7736455
34. Hasan, M.M., Rahman, M.M. A support vector machines based ransomware analysis framework with integrated feature set. In: 20th International Conference of Computer and Information Technology (ICCIT) (2017). https://doi.org/10.1109/ICCITECHN.2017.8281835