

Malware Classification using API System Calls

Allan Ninyesiga
aninyesiga@utamu.ac.ug
Uganda Technology and Management University

John Ngubiri
Uganda Technology and Management University

IJOTM
ISSN 2518-8623
DOI: 10.67610/ijotm.3.2.004
Volume 3. Issue II
p. 9, Dec 2018
http://jotm.utamu.ac.ug
email: ijotm@utamu.ac.ug

Abstract

Malware causes are increasing both in numbers and fatality. Hackers design malware to compromise systems security mostly confidentiality, integrity, and availability. Malware elimination techniques exist but the malware must be detected first. Malware detection techniques still have weaknesses of high false positive/ negatives rates. The emergency of polymorphic malware has made the situation worse. Recent studies have shown data mining to be promising in identifying malware by analysing API calls. However, in this approach, a file is detected as malicious or not. It is not classified on to which malware class it belongs. This makes its elimination harder as elimination schemes are mostly class based. Classification as a post detection process is important if the malware is to be eliminated from the system. We make an experimental study on use of data mining approach to classify malware using 4-gram API system calls. We use a dataset of 552 Windows Portable Executables (PE) with their corresponding API calls. The PE's were executed in a windows 7 virtual environment using the Cuckoo sandbox. Relevant 4-gram API call features are extracted using Term Frequency-Inverse Document Frequency (TF-IDF). Gaussian Naive Bayes, SVM, Random Forest, and Decision Trees were used to train and test the data. We show that the technique is successful with accuracy between 92% and 96.4%. There are internal variations in accuracy with SVM and Decision Trees performing best and Gaussian Naive Bayes performing worst.

Key words: *Malware, Portable Executable, API, Classification*

Introduction

Computers and Internet have become popular in day to day lives. They are actually part and parcel of many communities and businesses. The high use of Internet raised the degree of connectivity of electronic devices. For many systems therefore, the attack problem is rarely accessibility but rather accessibility. This puts in question the integrity of systems. Conventionally, software and computer systems are developed for good purpose. However, some software are developed to deliver malice (malware). A single code for example 'drop database' can delete terabytes of data in a single stroke. The I LOVE YOU virus, for example, caused damage worth $10 billion in ten days (Robert, 2016). Malware keeps a growing problem. Currently, it grows with more than 350,000 new instances daily (Malware, 2018).

Malware detection is traditionally done on susceptible files not process. This is largely by the signature, heuristic and behavioral approaches. Signature approach search for static patterns of known malware in suspicious files (Sulaiman & Ali, 2015). Studies have shown signature approach to be weak at exhausting

polymorphic and metamorphic malware. Heuristic approaches examine suspicious malware characteristics from suspicious files (Sulaiman & Ali, 2015). Despite of being able to detect unknown malware they suffer from high false positive rates. Behavioural approaches (Zahra, Hashem, Seyed, & Ali, 2013) monitors the program execution to identify suspicious behaviours. While it is able to detect different variants of malware, it also suffers from high false positives (Zarni & Win, 2013).

Detected malware is easily handled especially by elimination. However, the current polymorphic and metamorphic nature of malware makes them hard to detect by traditional means. They disguise their structure but not their operation. Since all malware have to execute to be successful, some studies (Hyun-il, 2016), (Youngjoon, Eunjin, & Huy, 2015) analysed API calls to detect malware in execution with high accuracy. However, this detection ends at flagging malware or not malware. It does not classify the malware into its types (virus, worm, Trojan, etc.). Classification is important as it helps in simplifying the course of action to neutralise it. We use 4-gram API calls to classify the detected malware.

The rest of the paper is organised as follows. Related work in Section 2. In section 3, we explain and justify the experimental setup and present results in Section 4. I then discuss the results in Section 5 and make conclusions and recommendations for future work in Section 6.

Related work

Attack strategies

Malware attack strategies can either be ordinary or network-based. Since the use of networks and internet is high, malware developers take the advantage to speed up the malicious attacks. These include Backdoors, Spyware, and Adware programs. On the other hand, Ordinary malware (like viruses) may not require network or internet to attack systems. They just need to be carried by the host program to another computer system. Malware usually attacks by corrupting computer system to affect the confidentiality, integrity of information and denial of services (Imtithal, Ali, & Ali, 2013). Some malware (such as a virus) creates replications which may exhaust computer resources (e.g. RAM, hard disk). Most new malware create variants through polymorphism and metamorphism making themselves evade detection by anti-malware programs (Kevadia, Prashant, & Nilesh, 2012).

Detection Approaches

Signature based approach maintains a database of signatures extracted from various known malware. Most of the antivirus engines around the world use this approach (Jyoti & Wankhade, 2013). Malware is normally detected by its signature pattern; for example, a cryptographic hash of a file, hash of file sections and/ series of bytes in a file (Sulaiman & Ali, 2015). A signature pattern is compared with the one stored in the database and if there is a match the file is detected malicious otherwise benign. Though signature based approach is good at detecting the known malware variants (Jyoti & Wankhade, 2013), it is unable to detect unknown malware variants (Zarni & Win, 2013). The approach also lacks the ability to detect polymorphic and metamorphic variants of malware (Zahra, Hashem, Seyed, & Ali, 2013).

Due to shortcomings of signature based detection mechanisms, some anti- malware developers resorted to using heuristics. Heuristic scanning approaches rely on rules and/or algorithms to look for commands, system behavior and keystrokes that may indicate malicious intent (Sulaiman & Ali, 2015). Without looking for specific signatures, heuristics scanning searches for certain commands or instructions within the program which are not found in a typical application programs (Lenny, n.d.). Though they try to detect unknown malware, their detection results yield to high false positives. This is because at lower granularity all software is made up of the same set of commands. A non-malicious command segment can easily be taken to be malicious and vice versa.

Behavioral approach observe behavior of a program to conclude whether it is malicious or not. Tools with behavior based mechanism seek to identify malware by monitoring for abnormal or suspicious behavior (Imtithal, Ali, & Ali, 2013). These behavior include; (i) attempt to alter host files, (ii) generation of autorun. inf files on removable media or on a network, (iii) sending of multiple mails, (iv) observing keystrokes, and or unpacking malicious code. However, studies show that behavior based detection approaches are susceptible to high false positive rates (Ashwini, Gayatri, & Meshram, 2013). The false positives are due to the fact that the malicious behaviour patterns can also be exhibited by benign programs.

The Role of Polymorphism and Metamorphism

Malware sustainability largely rely on ability to evade detection. Malware authors have tried to ensure that their malware is not detected by the antimalware engines. This is done by concealing their malware codes by polymorphism and metamorphism.

In polymorphism, a malware encrypts its self by an encrypting algorithm and a different key is used in any infection. In each execution therefore, a part of decryption code changes (Imtithal, Ali, & Ali, 2013) hence a different signature. This makes it hard to detect with anti-malware programs.

In metamorphism, malware change themselves in such a way that the new instance has a minimal or no resemblance to the original ones. Advanced detection techniques can easily detect their reliability by waiting for the virus to decrypt its self. Metamorphic viruses alter virus entire code but not changing the code's impact.

They can create variants of themselves using code-morphing and those morphed variants do not necessary have a common signature. These code changes makes it difficult for signature-based anti-malware software pro- grams to recognize that the different iterations are the same (Sanjam, Ekta, Divya, & Sanjeev, 2015).

API calls and Data Mining

Due to the shortcomings of traditional(signature, heuristics, and behavior) approaches of malware detection, researchers have shown that using data mining on program behavior features such as API calls can detect malware including metamorphic and polymorphic malware with high accuracy (Sanjam, Ekta, Divya, & Sanjeev, 2015), (Hamid, Mehdi, & Ahmad, 2014). This is because at a higher level, malware disguises themselves by changing their behaviour or continuously changing their signatures. However, to cause havoc they have to execute and changing the execution behaviour is harder. It may actually make them un-malicious/benign. This approach therefore targets malware at execution level.

(Hamid, Mehdi, & Ahmad, 2014), used a data mining approach to predict executable behavior using API that provides sequences captured of a running process. Results show the method is effective in detecting polymorphic and metamorphic mal- ware with the accuracy and detection rate of 93.5% and 95% respectively. The technique only classifies the program as malicious or benign. This is because at execution level, it is all about a process.

(Abhay, 2015), proposed a data mining approach to improve the malware detection ratio between malicious and benign with a high precision. Dark comet Trojan virus data was collected and processed using IDA pro as well as PEiD anti-packing reverse engineering tool. The ASM file generated by IDA pro was analyzed using machine learning techniques to examine shared malware patterns .The approach helped in detecting malware or benign basing on their code and obtained sequence of called system functions. The proposed approach can also detect obfuscated malware.

(Chun-I, Han-Wei, Chun-Han, & Yi-Fan, 2015), developed a hooking tool TraceHook to trace dynamic signatures that malware tries to hide in PE files. The tool traces code injections by intercepting the CreateThread function of a malicious process when PE file is being executed in a virtual environment. API call frequency is extracted as features and data mining algorithms (Nave Bayesian, J48, and SVM) are used to classify the behavior differences between malware and benign. Results show the method can achieve a high detection rate with low complexity by having a detection rate of 95% with only 80 attributes.

(Guanghui, Jianmin, & Chao, 2016), proposed a classification technique using dynamic analysis based on behavior profile. API system calls and other essential information of running malware are captured when the malware is running, then their multilayer dependency chain is established according to dependency relationship of these function calls. To identify the degree of similarity between malware variants, the similarity comparison algorithm is used.

(Hyun-il, 2016), proposed an approach to detecting malicious behaviors of software by analyzing information of API function calls. To recognize the malicious behavior of software, its behavior automaton is traced with a sequence of API function calls extracted during program execution. Malicious behavior can be identified by calculating similarity between the set of k-grams and a sequence of API function calls. (Youngjoon, Eunjin, & Huy, 2015), proposed an approach for dynamic analysis of malware by adopting DNA sequence alignment algorithms (Multiple Sequence Alignment and Longest Common Subsequences). The algorithms are used to extract common API call sequence patterns of malicious function from different categories of malware. Hooking process monitors are used to track the program's API call sequences when a new program needs to be traced. The sys- tem then compares the extracted API call sequences with API call sequence of API-based malware detection system database (APIMDS). If there is a match, APIMDS alerts the administrator.

(Zahra, Hashem, Seyed, & Ali, 2013), presented the use of features such as API system calls, OpCodes, N-Grams etc that can be used in methods to detect the behavior of malware. They show that using API system calls as a feature has some advantages which are;

> (i) they help in detecting polymorphic and unknown malware, (ii) outperforms other classification approaches in both detection ratio and accuracy, (iii) obfuscated malware variants can easily be detected, (iv) and help to detect malware before execution.

In this study, we adopt the use of API calls in classifying the malware by its behavior. Most of the related work detects malware by classifying it as malicious or benign. In our work we classify the malware by their behavior and specifying the class it belongs.

Experimental Process

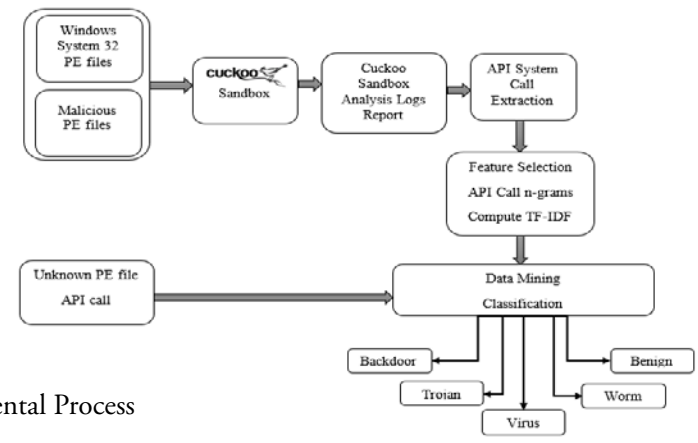


Fig. 1. The Experimental Process

Experimental Process

Shown in Fig. 1, the setup begins with the acquisition of Microsoft Windows PE files and malicious PE files. Benign PE files were extracted from a fresh installed Windows OS machine. Malicious PE files were collected from online malware repositories labelled by Kaspersky and VirusTotal (VirusTotal, n.d.), (Radu, Steven, & Thor, 2015).

The PE files were run in Cuckoo sandbox (Claudio, Alessandro, Jurriaan, & Mark, 2018) a malware analysis tool. The tool extracts API calls from the PE files during execution. The sandbox tool is configured in Ubuntu 14.04 alongside a windows7 virtual environment using oracle virtual box where the malicious and benign PE files were executed. The virtual environment help in such a way that malicious files execute and behave the same way like in normal system (Youngjoon, Eunjin, & Huy, 2015). This helps in understanding the behavior of malware when trying to infect the system.

During the PE file execution, Cuckoo sandbox generates log files. The log files contains the snapshots taken during execution (behavior profile). This is done for every sample that is executing.

Each API calls sequence is recorded in correspondence to its class label assigned by Kaspersky from VirusTotal (VirusTotal, n.d.). We consider four malware classes (Trojan, Virus, Backdoor, and Worm) and a benign software.

Feature Selection

The collected API call logs are always long and continuous which requires to break them up into ngrams. We apply data mining with TF-IDF (Jikku & P, 2015) feature selection technique to select relevant 4-gram API calls for classification. TF-IDF helps to identify a set of API calls that are more common in a malware/benign class. It works in a way that if the API call k-gram appears frequently in a class it is important and should be given a high score. But when it appears in too many other classes, it is not a unique identifier and should be assigned a lower score. Only the API call k-grams with a high score are considered to profile the behavior of a PE file.

Classification

After the feature selection process, data mining classification is applied using classification approaches. We used four classification approaches which include: SVM, Gaussian Naive Bayes, Random Forests, and Decision Trees. Basing on the kinds of API calls chosen to describe a certain class of malware/ benign, the classification approaches helps in concluding whether the file is benign or malicious by specifying which class on malware it belongs to. Since the technique process ends with specifying the class which the file belongs after the behavioral detection, malware mitigation can be simplified. Also since all PE have a direct linkage with the OS through API system calls, it indicates that the API calls can easily tell the malware behaviour when trying to execute.

Experimental Results

We run the experiment and classify the unknown malware/benign file. We analyse the results in areas of; (i) classifier accuracy, (ii) confusion matrix, (iii) Precision, Recall and F-Score measure and (iv) False Positive/ False Negative rates.

Classifier Accuracy

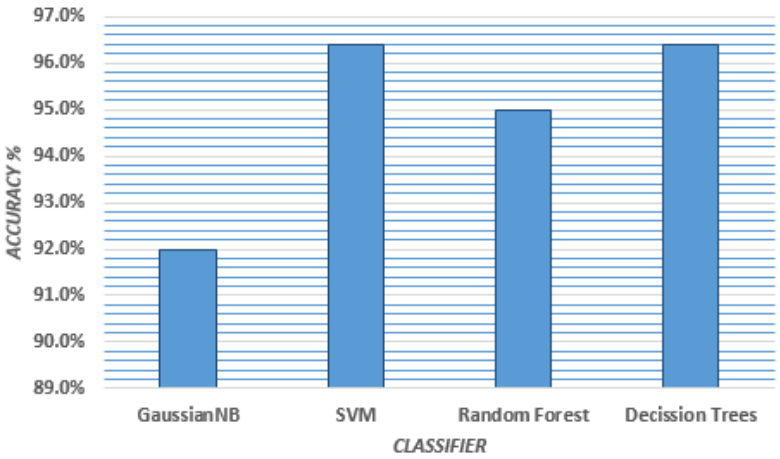


Fig. 2. Classifier Accuracy Results

Fig. 2 shows the performance accuracy results by classifiers used. The results indicate that SVM and Decision Trees performs better with the accuracy of 96.4% in classifying malware by its behaviour compared to other classifiers (Gaussian Naive Bayes & Random Forest). Despite relative variation in performance of classification approaches, the overall accuracy is high with the minimum being 92%. This shows that those approaches are promising approaches to classify malware identified during execution.

Confusion matrix

From Fig. 3, we observe that the detection rate is high with comparatively low false positive and negative rates. We further observe that: (i)Viruses are the easiest to detect among all malware, (ii)Backdoors are hard to detect, (iii)Trojans are easily confused into benign software by all classification approaches, and (iv) Trojans and Backdoors can easily be predicted as worms.

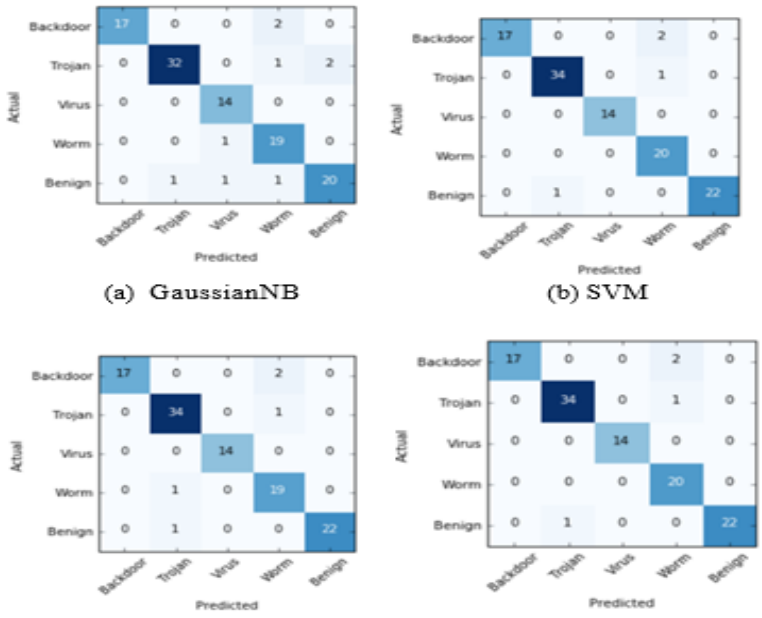


Fig. 3. Illustration of Confusion Matrices of Different Classification Model

Precision, Recall and F-Score Measures

Fig. 4 shows our approach yields a high precision, recall and F-Score results. These metrics results gives us high confidence that the accuracy results are correct. It also indicates that our classification algorithms performed well in classifying benign and malicious classes.

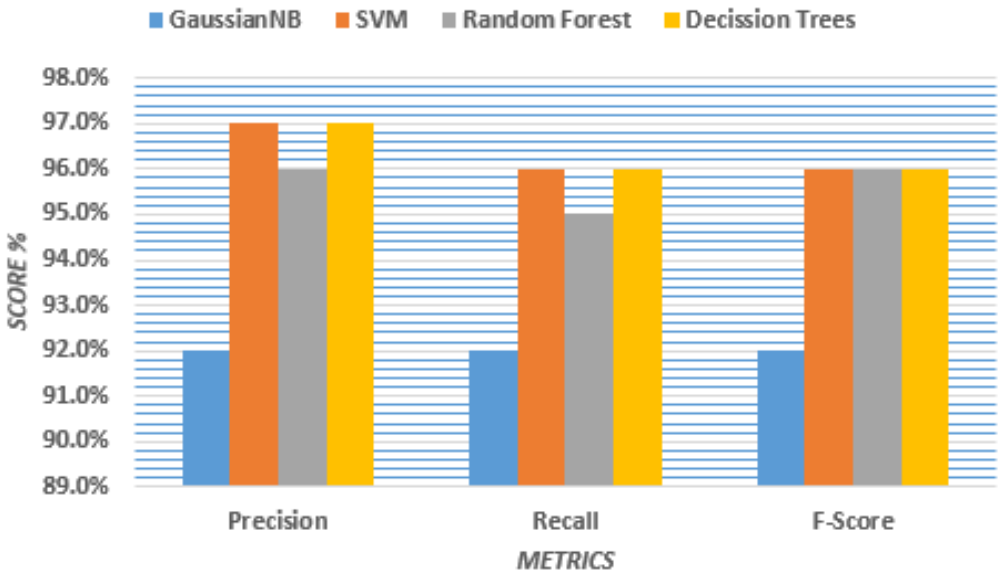


Fig. 4. Precision, Recall, and F-Score Metrics

False Positive and False Negative Rates

As another way of evaluating our technique, we calculate the False Positive Rate (FPR) and False Negative Rates (FNR) for each malware class and the benign class. Fig. 5 shows the rates are low with SVM and Decision Trees compared to other classification models. This proves our technique performs accurately. The virus class among all the classes was classified accurately by SVM, Random Forest, and decision Trees with 0 FPR and 0 FNR except for Gaussian Naive Bayes with a FPR of 0.02.

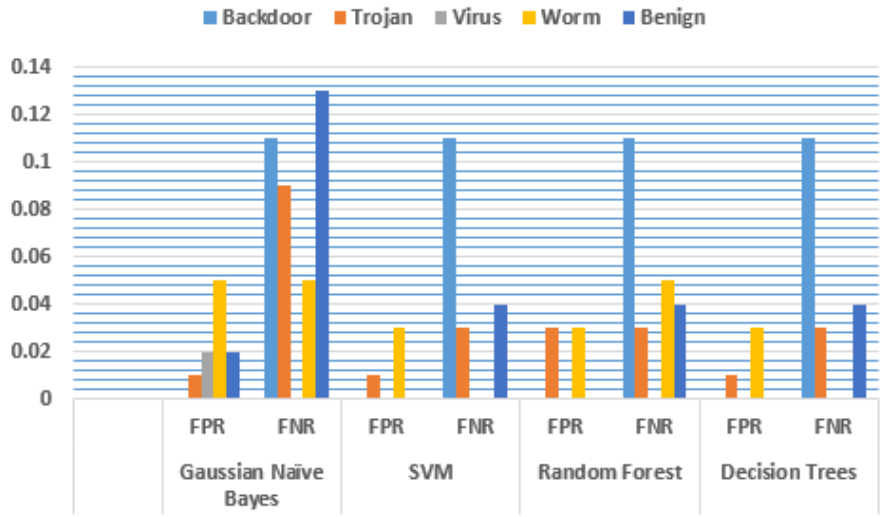


Fig. 5. Classification FPR's and FNR's

Discussion

Comparing the result evaluation by different used classification models, we observe that data mining API system calls gives good results in detecting and classifying malware. This is because all malware have a direct linkage by the OS either through embedding themselves to the software or by accessing them. Therefore hiding is impossible. Hiding at that level implies causing no harm. While they may get polymorphic that polymorphism is largely at application layer hence at interaction with the OS, the actions are less polymorphic. This implies that classification at API level is more robust and presents a more reliable approach to malware mitigation in systems.

Classifying malware by its behavior, we observe that SVM and Decision Trees classification models performs better than other models. From the con- fusion matrices results in Figure 3, we also observed key points that;(i) Viruses are easier to detect because their work largely deals with the OS rather than the files on which they are attached. In fact files are normally recovered when the antivirus addresses the issue. (ii) Backdoors, since they go through network ports they end up mixing with network instructions. They (network instructions) shield it making it harder to detect. (iii) Benign are predicted as Trojans in some cases. This is because Trojans tend to pretend to be useful programs. A useful program that can behave in a suspicious way is taken to be a Trojan. (iv)Backdoors also can easily be predicted as worms because they tend to use network ports to connect to victim's computer, and yet a number of worms usually spread across networks. This makes a Backdoor to be predicted as a worm during classification. Some Backdoors are usually carried by Trojans such that they can tend to hide them and tend to be useful programs. This too makes Trojans easily be predicted as worms.

Conclusion and Future work

Conclusion

We have done an experimental study using data mining to classify malware based on 4-gram API system calls as a behavioral feature. This was to extend previous studies that detect but not classify the malware. We used 552 malicious and benign PE files. Malicious PE's were collected from online repositories while benign from a fresh installed windows 7 OS. The PE's were executed in a virtual environment by the Cuckoo sandbox. We use four classifiers (Gaussian Naive Bayes, SVM, Rando Forest, and Decision Trees). The results show that mining API calls is a robust approach towards detecting malware by its behavior with (i) high accuracy, (ii) low false positive and (iii) false negative rates. SVM and Decision Tress are the best classifiers and the worst are Random Forest and Gaussian Naive Bayes. Viruses are the easiest to be detected and Backdoors are the hardest.

Future Work

In future, we hope to apply our technique on other various malware classes that were not considered in our research. And also evaluate our technique on a larger dataset of malware classes and benign PE's. Since our testing environment was Windows OS, we hope to apply the technique on Unix and MacOS systems. We also hope to automate the whole malware behavioral classification process.

References

Abhay, P. S. (2015). Improving the Malware Detection Ratio using Data Mining Techniques. *2nd International Conference on Science, Technology and Management* (pp. 852-857). Delhi: University of Delhi.

Ashwini, M., Gayatri, M., & Meshram, B. B. (2013). Analysis of Signature-Based and Behavior-Based Anti-Malware Approaches. *International Journal of Advanced Research in Computer Engineering and Technology (IJARCET)*, 2(6), 2037-2039.

Chun-I, F., Han-Wei, H., Chun-Han, C., & Yi-Fan, T. (2015). Malware Detection Systems Based on API Log Data Mining. *IEEE 39th Annual Computer Software and Applications Conference*. Taiwan: IEEE.

Claudio, G., Alessandro, T., Jurriaan, B., & Mark, S. (2018). *Cuckoo*. Retrieved from Cuckoo Sandbox: https://cuckoosandbox.org/

Guanghui, L., Jianmin, P., & Chao, D. (2016). A Behavior-Based Malware Variant Classification Technique. *International Journal of Information and Education Technology*, 6(4), 291-295.

Hamid, R. R., Mehdi, S., & Ahmad, K. (2014). A Novel Data Mining Method for Malware Detection. *Journal of Theoretical and Applied Information Technology*, 43-51.

Hyun-il, L. (2016). Detecting Malicious Behaviors of Software through Analysis of API Sequence k-grams. *Computer Science and Information Technology*, 4(3), 85-91. doi:10.13189/csit.2016.040301

Imtithal, S. A., Ali, S., & Ali, A. M. (2013). A Survey on Malware and Malware Detection Systems. *International Journal of Computer Applications*, 67(16), 0975-8887.

Jikku, K., & P, V. (2015). Unknown Metamorphic Malware Detection:Modelling with Fewer Relevant Features and Robust Feature Selection Techniques. *IAENG International Journal of Computer Science*.

Jyoti, L., & Wankhade, M. P. (2013, December). Malware and Malware Detection Techniques: A Survey. *International Journal of Engineering Research & Technology (IJERT)*, 2(12).

Kevadia, K., Prashant, S., & Nilesh, P. (2012). Metamorphic Malware Detection Using Statistical Analysis. *International Journal of Soft Computing and Engineering (IJSCE)*, 2(3).

Lenny, Z. (n.d.). *SearchSecurity*. Retrieved from https://searchsecurity.techtarget.com/tip/How-antivirus-software-works-Virus-detection-techniques

Malware. (2018). Retrieved from AV-TEST: https://www.av-test.org/en/statistics/malware/

Radu, P. S., Steven, H. S., & Thor, L. M. (2015). Analysis of Malware behavior: Type classification using machine learning. *2015 International Conference on Cyber Situational Awareness, Data Analytics and Assessment* (CyberSA).

Robert, K. (2016, March 8). *15 Most Dangerous Malware Of All Time*. Retrieved from zoo computer repairs: https://www.zoorepairs.com.au/computer-tips/list-of-most-dangerous-malware-of-all-time/

Sanjam, S., Ekta, G., Divya, B., & Sanjeev, S. (2015). Detecting and Classifying Morphed Malwares: A Survey. *International Journal of Computer Applications* (, 122(10), 0975-8887.

Sulaiman, A. A., & Ali, A. (2015). A Comparative Study of Virus Detection Techniques. *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, 9(6), 1566-1573.

VirusTotal. (n.d.). Retrieved July 12, 2018, from VirusToatal Website: https://www.virustotal.com/#/home/upload

Youngjoon, K., Eunjin, K., & Huy, K. K. (2015). A Novel Approach to Detect Malware Based on API Call Sequence Analysis. *International Journal of Distributed Sensor Networks*, 11(6).

Zahra, B., Hashem, H., Seyed, M. H., & Ali, H. (2013). A Survey on Heuristic Malware Detection Techniques. *5th Conference on Information and Knowledge Technology (IKT)* (pp. 113-120). Shiraz Iran : IEEE.

Zarni, A., & Win, Z. (2013). Permission-Based Android Malware Detection. *International Journal of Scientific & Technology Research*, 2(3), 228-234.