INTERNATIONAL JOURNAL
ON INFORMATICS VISUALIZATION

journal homepage : www.joiv.org/index.php/joiv

Dynamic Ransomware Detection for Windows Platform Using Machine Learning Classifiers

M. Izham Jaya [a], Mohd Faizal Ab. Razak [a,*]

[a] *Faculty of Computing, College of Computing and Applied Sciences, Universiti Malaysia Pahang, 26600 Pekan, Pahang, Malaysia*
Corresponding author: [*]faizalrazak@ump.edu.my

***Abstract*— Ransomware attacks are also rising in this growing technological advancement world. This threat often affects the finance of individuals, organizations, and financial sectors. To effectively detect and block these ransomware threats, the dynamic analysis strategy was proposed and carried out as the approach of this research. This paper aims to detect ransomware attacks with dynamic analysis and classify the attacks using various machine learning classifiers: Random Forest, Naïve Bayes, J48, Decision Table, and Hoeffding Trees. The TON IoT Datasets from the University of New South Wales (UNSW) were used to capture ransomware attack features on Windows 7. During the experiment, a testbed was configured with numerous virtual Windows 7 machines and a single attacker host to carry out the ransomware attack. Seventy-seven classification features are selected based on the changes before and after the attack. Random Forest and J48 classifiers outperformed other classifiers with the highest accuracy results of 99.74%. The confusion matrix highlights that both Random Forest and J48 classifiers can accurately classify the ransomware attacks with the AUC value of 0.997, respectively. Our experimental result also suggests that dynamic analysis with a machine learning classifier is an effective solution to detect ransomware with an accuracy percentage exceeding 98%.**

***Keywords*— Malware; ransomware detection; machine learning; classifier.**

Manuscript received 22 Dec. 2021; revised 29 Jan. 2022; accepted 10 Apr. 2022. Date of publication 31 Aug. 2022.

I. INTRODUCTION

Currently, attackers employ sophisticated methods to create new types of profitable malware. Ransomware is a type of cyber-attack that has recently gained popularity. The goal of this malware is to encrypt user files, restrict access to them, and then demand a ransom for the decryption key [1], [2]. Ransomware has become a serious menace to the computing sector, necessitating fast action to avoid financial and moral extortion. As a result, a new technique to detect and prevent this type of assault is critical. Dynamic analysis, static analysis, and a hybrid system that combined dynamic and static analysis were the three types of detection approaches used [3]–[5]. Most previous detection methods relied on a time-consuming but feasible and effective procedure known as dynamic analysis [6], [7].

Ransomware attacks first appeared in September 2013 using Rivest–Shamir–Adleman (RSA) public-key cryptography. When more than 1,400,000 Kaspersky users across numerous industries were targeted in 2016, it escalated into a catastrophic problem. Since the ransomware "WannaCry" infected over 400,000 machines across 150 countries in just one day in 2017, researchers have concentrated their efforts on ransomware detection. Despite this, the number of ransomware victims among enterprises globally has climbed during 2018, reaching a high of 68.5 percent in 2021 [8]. Figure 1 depicts the global business victimization rate from 2018 to 2021.

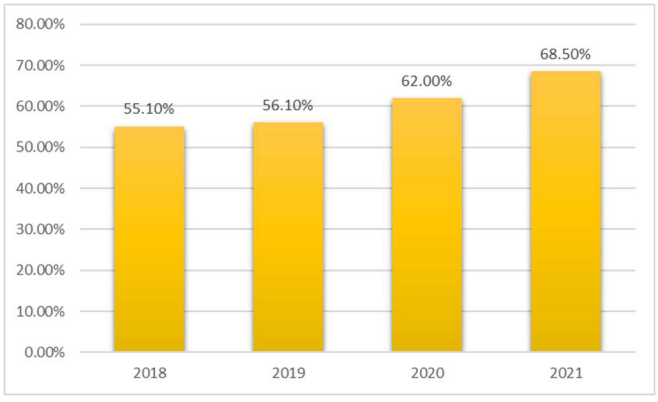

Fig. 1 Number of ransomware victimization in businesses worldwide

Over the years, various machine learning techniques and frameworks for ransomware detection have been developed and tested. According to multiple research studies, ransomware detection rates can exceed 96 percent when a machine learning algorithm combined with a dynamic analysis approach is used [9], [10]. The machine learning algorithm can also be used to analyze network traffic for Android malware, with detection rates exceeding 99 percent[11]. Due to this, the focus of this research is on detecting ransomware attacks on a device machine; the following are the primary contributions of this research:

- The detection of ransomware with dynamic analysis of before and after the attacks takes place using different types of machine learning classifiers.
- The comparison of accuracy between different machine learning classifiers on finding the ransomware attacks on the Windows 7 machine.
- The Random Forest and J48 classifiers have proven to be the most accurate implementations in determining a ransomware attack on a machine.

Omar et al. [12] discussed that a ransomware-affected Windows 7 machine has distinct network dialogues establishment characteristics. The infected machine will connect to a remote attacker's network address, a command-and-control server, and a payment or distribution website. Due to this reason, we gathered a set of normal feature behaviors on the Windows 7 machine prior to ransomware infection and classified the ransomware-affected Windows 7 features using various classification approaches to determine the attacks.

Previously published research by Takeuchi et al. [13] used the Support Vector Machines (SVM) algorithm to detect ransomware. SVM is a supervised machine learning algorithm, and the approach aims to train the SVM to recognize the API calls as ransomware detection features. The researchers conducted a testbed study by analyzing 276 types of ransomwares, including WannaCry, PETYA, and CryptoLocker in the Cuckoo Sandbox. The SVM detection rate has been shown to be higher than the findings by Rieck et al. [14], with ransomware detection accuracy results of 97.48 percent and a missing rate of 1.64 percent, which is lower than the reported missing rate Rieck et al. [14]. For the unknown ransomware to be properly recognized, the SVM approach implemented by Takeuchi et al. [13] thoroughly examines the API call sequences, and the authors' vector representations include the number of q-grams in the execution logs. As a result, ransomware is less likely to go undetected. As proposed by Takeuchi et al. [13], dynamic analysis for ransomware detection tends to produce high accuracy results as it is more difficult for the malware to hide its behavior than to hide its underlying code.

A ransomware attack can also be detected using the honeypot approach. The honeypot strategy entails network administrators establishing phony computer resources that can be used as decoy machines to detect suspicious behavior [15]. The honeypot approach allows for real-time monitoring of ransomware activity and attack methods. However, such an approach could backfire if it is used as a launchpad to infect other parts of the system. A honeypot approach is employed by Pavithra and Selvakumara Samy [16], with a bogus folder created to funnel the attack and monitor the changes in real-time. Other researchers have developed tools and applications to detect ransomware attacks based on this concept. Paybreak, a specific mechanism for storing cryptographic encryption keys in a key vault, is proposed by Kolodenker et al. [17]. It is then used to decrypt the files and folders encrypted by ransomware. SH-VARR is a framework developed by Al-Dwairi et al. [18], which utilizes the link concept to protect any XML document from being encrypted or deleted during a ransomware attack.

Identifying different types of ransomwares is a difficult and time-consuming task. It is becoming more difficult to establish a solid overcoming method to deal with ransomware attacks since ransomware creators are constantly upgrading their products to avoid any new detection methods. To overcome this, researchers employed Software-Defined Networking (SDN) for ransomware detection using deep packet tracing with POST and GET requests [19], [20]. Once the ransomware is detected, the IP addresses of the servers will be blacklisted by the determined servers in charge of controlling the addresses. Alas, this countermeasure method has a high false-positive rate of nearly 4.95 percent, which makes it a cause of faulty and incorrectly constrained useful services [20]. As a result, the network administrator must check the victim's computer's network traffic and the server for unlawful contact and prohibit the encryption key transfer.

A classification model can be implemented to analyze a victim's computer traffic and to identify the encryption key retrieval process. As such, EldeRAN is proposed by Sgandurra et al. [21], which delivers a highly accurate ransomware dynamic analysis using a machine learning algorithm. The EldeRAN is designed to identify ransomware infestations with a significantly higher true-positive and a low false-positive rate. The results determined that the API calls and the registry key are important in determining the most relevant classification features. EldeRAN can also identify unknown ransomware, with an average error rate of 2.4 percent.

II. Materials and Method

Fig. 2 describes five phases in conducting this research: the literature review process, dataset collection, analysis of relevant features for ransomware detection on Windows 7, development of the classification model, model testing, and comparing the experiment results.

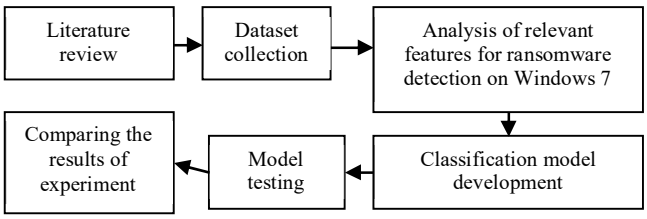

Fig. 2 Components flow of the ransomware detection system

This research presents an in-depth analysis of the ransomware attack detection using dynamic analysis and applied various machine learning classifiers to classify the ransomware attack: Random Forest, Naïve Bayes, J48, Decision Table, and Hoeffding Trees. The classification features are derived from comparing Windows 7 operating system features before and after the ransomware outbreak,

and the features change is observed and used to characterize the ransomware attack on the Windows 7 machine.

A. Dataset Collection

The TON IoT Datasets [22] from the University of New South Wales' (UNSW) were used to capture ransomware attack features on Windows 7. A total of 132 ransomware attack features that targeted Windows 7 machine have been taken from the database. This data was acquired primarily through a realistic and large-scale network built at the Australian Defense Force Academy's Cyber Range and IoT Labs (ADFA), Canberra's School of Engineering and Information Technology (SEIT), UNSW. The dataset is crucial for ensuring that the classification results are accurate. On the other hand, the dataset assists the researchers in better understanding ransomware features and explaining the behavior of ransomware outbreaks on the Windows 7 operating system.

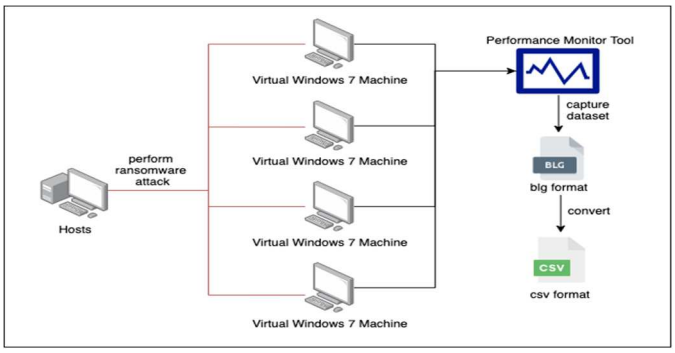

Fig. 3 Ransomware attacks testbed

The researcher then investigates the dataset further, with the results used to classify potential ransomware attacks. The testbed was configured with numerous virtual Windows 7 machines and a single attacker host to carry out the ransomware attack, as illustrated in Fig. 3.

The Performance Monitor Tool was used to collect the data that details the attack. The raw data was collected in a *.blg file containing data for desk, process, processor, memory, and network activities. The status of normal dataset features before the ransomware attack has been labelled "Normal," while the successfully infected machine has been labelled "Ransomware."

B. Machine Learning Approach for the Ransomware Classification

Waikato Environment for Knowledge Analysis (WEKA) tool is used to implement feature selection and machine learning classification algorithms in this research. It is a well-known Java-based machine learning software created at New Zealand's Waikato University [23], [24]. WEKA can implement a wide range of data mining tasks, including data preprocessing, clustering, classification, regression, visualization, and feature selection [25]–[27].

The Windows 7 features listed in Table 1 are categorized based on key attributes and used to train the classification model. The collected features are optimized using feature optimization to ensure accurate ransomware attack classification [28]. This approach streamlines the ransomware classification process by reducing training and testing time during the classification. Irrelevant and redundant traits from the dataset that do not add to the classification model's accuracy are determined and eliminated using the feature selection method [29]. The total number of selected features is reduced from 132 to 77. The reduction is based on the fact that certain features are not significantly changed after the ransomware attack.

The dataset with the selected features will be used to train the ransomware classification model and to identify the ransomware attack during the testing phase. Thus, the dataset is divided into two portions at a ratio of 70:30.

TABLE I
WINDOWS 7 FEATURES

Features	
Processor_DPC_Rate	Process_pct_User_Time
Processor_pct_idle_Time	Process_Virtual_Bytes_Peak
Processor_pct_interupt_Time	Process_Page_Files_Bytes_Peak
Processor_pct_User_Time	Process_IO_Other_Bytes_sec
Processor_pct_C1_Time	Process_Privates_Bytes
Processor_pct_Processor_Time	Process_IO_Write_Bytes_sec
Processor_C1_transition_sec	Process_Virtual_Bytes
Processor_pct_DPC_Time	Process_pct_Processor_Time
Processor_pct_Privilleged_Time	Process_Pool_Nonpaged_Bytes
Processor_DPCs_Queued_sec	Process_Working_set
Processor_interupts_sec	Process_Page_Faults_sec
Process_pool_Paged_bytes	Process_IO_Other_Operations_sec
Process_IO Read_Operations_sec	Process_IO_Data_Operations_sec
Process_Handle_count	Process_Thread_Count
Network_I(Intel[R]_Pro_1000MT)_Bytes_Received_sec	Process_pct_privilleged_time
Network_I(Intel[R]_Pro_1000MT)_Bytes_Sent_sec	Process_IO_Data_Bytes_sec
Network_I(Intel[R]_Pro_1000MT)_Bytes_Total_sec	Process_IO_Read_Bytes_ sec
Network_I(Intel[R]_Pro_1000MT)_packets_Received_sec	Memory__System_Code_Resident_Bytes
Network_I(Intel[R]_Pro_1000MT)_packets_Sent_sec	Memory_Available_Bytes
Network_I(Intel[R]_Pro_1000MT)_Packets_sec	Memory_Commit_Limit
Network_I(Intel[R]_Pro_1000MT)_Packets_Sent_sec	Memory_Transition_Pages_RePurposed_sec
Network_I(Intel[R]_Pro_1000MT)_Peackets_Received_sec	Memory_Pages_Output_sec
Memory_Pool_Page_bytes	Memory_Page_Reads_sec

Memory_Free&Zero_Page_List_bytes	Memory_Demands_Zero_Faults_sec
Memory_Caches_bytes_Peak	Memory_Available_Kbytes
Process_Working_Set_Peak	Memory_Pages_sec
Process_IO_WriteOperations_sec	Memory_Cache_Bytes
Process_Page_File_bytes	Memory_Pool_Nonpages_bytes
Memory_Pool_Paged_Allocs	Memory_Page_Faults_sec
Memory_Pool_Nonpaged_Allocs	Memory_Transition_Faults_sec
Memory_pct_Commited_Bytes_In_Use	Memory_System_Cache_Resident_Bytes
Memory_Free_System_Page_able_Entries	Memory_Standby_Cache_Reserves_Bytes
Memory_Available_Mbytes	Memory_Page_Writes_Sec
Memory_Modified_Page_List_Bytes	Memory_Standby_Cache_Core_Bytes
Memory_Cache_Faults_sec	Memory_System_Driver_Resident_Bytes
Memory_Committed_Bytes	Memory_Standby_Cache_Normal_Priority_Bytes
Memory_System_Driver_total_Bytes	Memory_Pool_Paged_Resident_Bytes
Memory_Pages_Input_sec	Memory_Write_Copies_sec
Features	
Processor_DPC_Rate	Process_pct_User_Time
Processor_pct_idle_Time	Process_Virtual_Bytes_Peak
Processor_pct_interupt_Time	Process_Page_Files_Bytes_Peak
Processor_pct_User_Time	Process_IO_Other_Bytes_sec
Processor_pct_C1_Time	Process_Privates_Bytes
Processor_pct_Processor_Time	Process_IO_Write_Bytes_sec
Processor_C1_transition_sec	Process_Virtual_Bytes
Processor_pct_DPC_Time	Process_pct_Processor_Time
Processor_pct_Privilleged_Time	Process_Pool_Nonpaged_Bytes
Processor_DPCs_Queued_sec	Process_Working_set
Processor_interupts_sec	Process_Page_Faults_sec
Process_pool_Paged_bytes	Process_IO_Other_Operations_sec
Process_IO Read_Operations_sec	Process_IO_Data_Operations_sec
Process_Handle_count	Process_Thread_Count
Network_I(Intel[R]_Pro_1000MT)_Bytes_Received_sec	Process_pct_privilleged_time
Network_I(Intel[R]_Pro_1000MT)_Bytes_Sent_sec	Process_IO_Data_Bytes_sec
Network_I(Intel[R]_Pro_1000MT)_Bytes_Total_sec	Process_IO_Read_Bytes_ sec
Network_I(Intel[R]_Pro_1000MT)_packets_Received_sec	Memory__System_Code_Resident_Bytes
Network_I(Intel[R]_Pro_1000MT)_packets_Sent_sec	Memory_Available_Bytes
Network_I(Intel[R]_Pro_1000MT)_Packets_sec	Memory_Commit_Limit
Network_I(Intel[R]_Pro_1000MT)_Packets_Sent_sec	Memory_Transition_Pages_RePurposed_sec
Network_I(Intel[R]_Pro_1000MT)_Peackets_Received_sec	Memory_Pages_Output_sec
Memory_Pool_Page_bytes	Memory_Page_Reads_sec
Memory_Free&Zero_Page_List_bytes	Memory_Demands_Zero_Faults_sec
Memory_Caches_bytes_Peak	Memory_Available_Kbytes
Process_Working_Set_Peak	Memory_Pages_sec
Process_IO_WriteOperations_sec	Memory_Cache_Bytes
Process_Page_File_bytes	Memory_Pool_Nonpages_bytes
Memory_Pool_Paged_Allocs	Memory_Page_Faults_sec
Memory_Pool_Nonpaged_Allocs	Memory_Transition_Faults_sec
Memory_pct_Commited_Bytes_In_Use	Memory_System_Cache_Resident_Bytes
Memory_Free_System_Page_able_Entries	Memory_Standby_Cache_Reserves_Bytes
Memory_Available_Mbytes	Memory_Page_Writes_Sec
Memory_Modified_Page_List_Bytes	Memory_Standby_Cache_Core_Bytes
Memory_Cache_Faults_sec	Memory_System_Driver_Resident_Bytes
Memory_Committed_Bytes	Memory_Standby_Cache_Normal_Priority_Bytes
Memory_System_Driver_total_Bytes	Memory_Pool_Paged_Resident_Bytes
Memory_Pages_Input_sec	Memory_Write_Copies_sec

III. Result and Discussion

The supervised machine learning approach was used in this research as it provides a promising outcome by reducing errors [30], [31]. This research compares the performance of various notable classifiers such as Random Forest, Naive Bayes, J48, Decision Table, and Hoeffding Trees using five different classifiers. The accuracy percentage, false positive rate (FPR), precision, recall, and the F-Measure metrics are utilized to evaluate the research. Table 2 lists all five classifiers' results in the testing phase.

TABLE II
PERFORMANCE OF EACH CLASSIFIER

Classifier	Accuracy (%)	FPR	Precision (%)	Recall (%)	F-Measure (%)
Random Forest	99.74	0.26	99.70	99.70	99.70
Naïve Bayes	99.48	0.52	99.50	99.50	99.50
J48	99.74	0.26	99.70	99.70	99.70
Decision Table	98.70	1.30	98.70	98.70	98.70
Hoeffding Tree	99.48	0.52	99.50	99.50	99.50

Compared to the other classifiers, Random Forest and J48 hold the highest accuracy of 99.74 percent. Decision Table, on the other hand, had the lowest accuracy percentage of 98.70 percent. The results indicate that the Random Forest and J48 classifiers accurately identify ransomware outbreaks. The high accuracy percentage in both Random Forest and J48 classifiers is backed up by a high precision percentage of 97.70 percent. Furthermore, the high accuracy percentage obtained in the result may indicate that the feature selection strategy used in this research had an important part in producing superior results.

A. Confusion Matrix

The confusion matrix is used to summarize the performance of the classification model. Table 3 presents the results of two types of classification: normal and ransomware. For all classifiers, the result will be labeled as "ransomware" when the classification model anticipates the presence of ransomware activity and vice versa.

J48 and Random Forest outperformed other classifiers in ransomware detection when all 82 actual ransomwares were classified as "ransomware". In terms of false predictions, both J48 and Random Forest classifiers hold the lowest number with only 1 case. Therefore, the J48 and Random Forest classifiers are more accurate in detecting ransomware attacks than the other classifiers.

TABLE III
CONFUSION MATRIX OF EACH CLASSIFIER

Classifier	Actual	Classification	
		Classified "Normal"	Classified "Ransomware"
Random Forest	Actual normal	301	1
	Actual ransomware	0	82
Naïve Bayes	Actual normal	301	1
	Actual ransomware	1	81
J48	Actual normal	301	1
	Actual ransomware	0	82
Decision Table	Actual normal	300	2
	Actual ransomware	3	79
Hoeffding Tree	Actual normal	301	1
	Actual ransomware	1	81

B. Receiver Operating Characteristics (ROC) Curve

In addition to the performance matrix, the ROC curve for each machine learning classifier was calculated in this research. True Positive Rate (TPR) was selected as the detection rate that accurately classified the ransomware attack, whereas FPR was chosen as the detection rate that incorrectly classified the normal cases as a ransomware attack. The ROC curve is shown in Fig. 4.

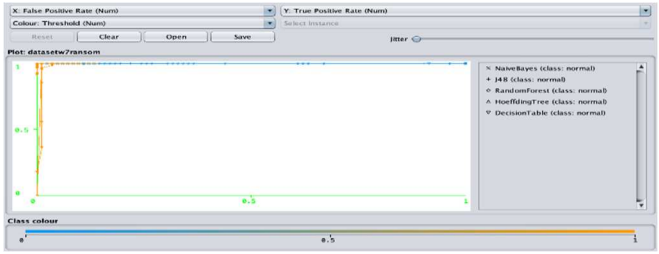

Fig. 4 ROC curve

The horizontal axis of Fig. 4 depicts the error detection rate, whereas the vertical axis depicts the detection rate. The four lines represent the ROC curves of the machine learning classifier. The ROC curves are difficult to compare under the same conditions. As a result, the bottom portion of the curve was used to calculate the recognition accuracy (AUC). The AUC results can be used to establish if the classifiers are effective in detecting ransomware or not. There are two color ranges in the class. The perfect classification is represented by a range of 0.5 to 1, whereas a range of 0 to 0.5 represents an inadequate classification. According to the AUC results in Table 4, Random Forest and J48 classifiers possess the highest AUC values of 0.997. The outcome implies that both classifiers performed admirably. The ROC curve and AUC values given in this research demonstrated that all classifiers produced compelling and accurate results in detecting ransomware outbreaks.

TABLE IV
AUC RESULTS

Classifier	AUC	Prediction
Random Forest	0.997	Perfect prediction
Naïve Bayes	0.993	Perfect prediction
J48	0.997	Perfect prediction
Decision Table	0.987	Perfect prediction
Hoeffding Tree	0.992	Perfect prediction

Additionally, Table 5 compares the time taken to train the classification model. The result shows that the Naïve Bayes classifier holds the fastest training time. Random Forest and J48 came in second and third, respectively, with the Hoeffding Tree trailing by a small margin. The result also suggests that the Decision Tree takes the longest time to train the classification model.

TABLE V
TIME TAKEN TO BUILD MODEL (SECONDS)

Classifier	AUC
Random Forest	0.07
Naïve Bayes	0.05
J48	0.07
Decision Table	0.46
Hoeffding Tree	0.09

IV. CONCLUSION

This research examined the performance of five machine learning classifiers in detecting ransomware attacks: Random Forest, Naïve Bayes, J48, Decision Table, and Hoeffding Trees. A dynamic analysis approach that implements machine learning classifiers is used in this research, and the ransomware attacks were accurately classified according to the selected features. The research analyses the sample dataset and the Windows 7 engine to monitor changes in the features before and after the ransomware attack. The experiment's findings demonstrated that the Random Forest and J48 classifiers had the highest percentage of accuracy in detecting a ransomware attack.

Based on the findings, the following conclusions can be drawn: The results show that the Random Forest and J48 classifier with dynamic analysis can detect ransomware attacks with remarkable performance. The findings highlight that the Random Forest tree size of 77 has produced a high accuracy of 97.74 percent, a high ROC of 99.7 percent, and a

low FPR of 0.26 with just 0.07 seconds of dataset training time. The reduction of classification features from 78 to 191 positively improved the ransomware detection rate. The Random Forest classifier performed similarly to J48 in all tests, resulting in the most effective solution to detect ransomware attacks.

ACKNOWLEDGMENT

The researchers thank the Universiti Malaysia Pahang for the financial support under the UMP Internal Research Grant RDU200317. This support is gratefully acknowledged.

REFERENCES

[1] J. Zhu, J. Jang-Jaccard, A. Singh, I. Welch, H. AI-Sahaf, and S. Camtepe, "A few-shot meta-learning based Siamese neural network using entropy features for ransomware classification," *Computers and Security*, vol. 117, Jun. 2022, doi: 10.1016/J.COSE.2022.102691.

[2] M. Mohd Azuwan Efendy, A. R. Mohd Faizal, and A. R. Munirah, "Malware Detection System Using Cloud Sandbox, Machine Learning," International Journal of Software Engineering and Computer Systems, vol. 8, no. 2, pp. 25–32, Jul. 2022, doi: 10.15282/IJSECS.8.2.2022.3.0100.

[3] R. Almohaini, I. Almomani, and A. Alkhayer, "Hybrid-Based Analysis Impact on Ransomware Detection for Android Systems," Applied Sciences, vol. 11, no. 22, p. 10976, Nov. 2021, doi: 10.3390/APP112210976.

[4] S. Mahdavifar, D. Alhadidi, and A. A. Ghorbani, "Effective and Efficient Hybrid Android Malware Classification Using Pseudo-Label Stacked Auto-Encoder," Journal of Network and Systems Management, vol. 30, p. 22, 2022, doi: 10.1007/s10922-021-09634-4.

[5] I. Kara and M. Aydos, "The rise of ransomware: Forensic analysis for windows based ransomware attacks," Expert Systems with Applications, vol. 190, p. 116198, Mar. 2022, doi: 10.1016/J.ESWA.2021.116198.

[6] Y.-T. HuangID, Y. S. Sun, and M. Chang Chen, "TagSeq: Malicious behavior discovery using dynamic analysis," PLOS ONE, vol. 17, no. 5, p. e0263644, May 2022, doi: 10.1371/JOURNAL.PONE.0263644.

[7] G. McDonald, P. Papadopoulos, N. Pitropakis, J. Ahmad, and W. J. Buchanan, "Ransomware: Analysing the Impact on Windows Active Directory Domain Services," Sensors 2022, Vol. 22, Page 953, vol. 22, no. 3, p. 953, Jan. 2022, doi: 10.3390/S22030953.

[8] Statista Research Department, "Global ransomware victimization rate since 2018 to 2021," Statista Research Department. https://www.statista.com/statistics/204457/businesses-ransomware-attack-rate/ (accessed Jul. 18, 2022).

[9] J. Singh and J. Singh, "Assessment of supervised machine learning algorithms using dynamic API calls for malware detection," International Journal of Computers and Applications, vol. 44, no. 3, pp. 270–277, 2022, doi: 10.1080/1206212X.2020.1732641.

[10] J. Palša et al., "MLMD-A Malware-Detecting Antivirus Tool Based on the XGBoost Machine Learning Algorithm," Applied Sciences, vol. 12, no. 13, p. 6672, Jul. 2022, doi: 10.3390/APP12136672.

[11] J. Mohamad Arif, M. F. Ab Razak, S. R. Tuan Mat, S. Awang, N. S. N. Ismail, and A. Firdaus, "Android mobile malware detection using fuzzy AHP," Journal of Information Security and Applications, vol. 61, p. 102929, Sep. 2021, doi: 10.1016/J.JISA.2021.102929.

[12] Omar M.K. Alhawi, James Baldwin, and A. Dehghantanha, "Leveraging Machine Learning Techniques for Windows Ransomware Network Traffic Detection," in Cyber Threat Intelligence. Advances in Information Security, vol. 70, A. Dehghantanha, M. Conti, and T. Dargahi, Eds. Springer, Cham, 2018. doi: 10.1007/978-3-319-73951-9_5.

[13] Y. Takeuchi, K. Sakai, and S. Fukumoto, "Detecting ransomware using support vector machines," in ICPP '18: Proceedings of the 47th International Conference on Parallel Processing Companion, Aug. 2018, vol. 1. doi: 10.1145/3229710.3229726.

[14] K. Rieck, P. Trinius, C. Willems, and T. Holz, "Automatic analysis of malware behavior using machine learning," Journal of Computer Security, vol. 19, no. 4, pp. 639–668, Jan. 2011, doi: 10.3233/JCS-2010-0410.

[15] S. Touch and J.-N. Colin, "A Comparison of an Adaptive Self-Guarded Honeypot with Conventional Honeypots," Applied Sciences, vol. 12, no. 10, p. 5224, May 2022, doi: 10.3390/APP12105224.

[16] J. Pavithra and S. Selvakumara Samy, "A Comparative Study on Detection of Malware and Benign on the Internet Using Machine Learning Classifiers," Mathematical Problems in Engineering, vol. 2022, pp. 1–8, Jun. 2022, doi: 10.1155/2022/4893390.

[17] E. Kolodenker, W. Koch, G. Stringhini, and M. Egele, "PayBreak: Defense Against Cryptographic Ransomware," in Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, 2017, pp. 599–611. doi: 10.1145/3052973.3053035.

[18] M. Al-Dwairi, A. S. Shatnawi, O. Al-Khaleel, and B. Al-Duwairi, "Ransomware-Resilient Self-Healing XML Documents," Future Internet, vol. 14, no. 4, p. 115, Apr. 2022, doi: 10.3390/FI14040115.

[19] H.-M. Chuang, F. Liu, and C.-H. Tsai, "Early Detection of Abnormal Attacks in Software-Defined Networking Using Machine Learning Approaches," Symmetry (Basel), vol. 14, no. 6, p. 1178, Jun. 2022, doi: 10.3390/SYM14061178.

[20] G. Cusack, O. Michel, and E. Keller, "Machine Learning-Based Detection of Ransomware Using SDN," in Proceedings of the 2018 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization, 2018, vol. 18, pp. 1–6. doi: 10.1145/3180465.3180467.

[21] D. Sgandurra, L. Muñoz-González, R. Mohsen, and E. C. Lupu, "Automated Dynamic Analysis of Ransomware: Benefits, Limitations and use for Detection," Cryptography and Security, arXiv 2016, Accessed: Jul. 18, 2022. [Online]. Available: https://arxiv.org/abs/1609.03020v1

[22] "The TON_IoT Datasets | UNSW Research," UNSW Canberra at ADFA, 2021. https://research.unsw.edu.au/projects/toniot-datasets (accessed Jul. 18, 2022).

[23] S. Madugula, S. Kiran, V. Chandra Shekhar Rao, S. Venkatramulu, M. S. B. Phridviraj, and S. Pratapagiri, "Advanced Machine Learning Scenarios for Real World Applications using Weka Platform," Proceedings of the International Conference on Electronics and Renewable Systems, ICEARS 2022, pp. 1215–1218, 2022, doi: 10.1109/ICEARS53579.2022.9752368.

[24] N. Thushika and S. Premaratne, "A Data Mining Approach for Parameter Optimization in Weather Prediction," International Journal of Data Science, vol. 1, no. 1, pp. 1–13, Apr. 2020, doi: 10.18517/IJODS.1.1.1-13.2020.

[25] A. A. R. Melvin et al., "Dynamic malware attack dataset leveraging virtual machine monitor audit data for the detection of intrusions in cloud," Transactions on Emerging Telecommunications Technologies, vol. 33, no. 4, p. e4287, Apr. 2022, doi: 10.1002/ETT.4287.

[26] J. Pattee, S. M. Anik, and B. K. Lee, "Performance Monitoring Counter Based Intelligent Malware Detection and Design Alternatives," IEEE Access, vol. 10, pp. 28685–28692, 2022, doi: 10.1109/ACCESS.2022.3157812.

[27] D. W. Fernando and N. Komninos, "FeSA: Feature selection architecture for ransomware detection under concept drift," Computers & Security, vol. 116, p. 102659, May 2022, doi: 10.1016/J.COSE.2022.102659.

[28] M. Sulistiyono, L. A. Wirasakti, and Y. Pristyanto, "The Effect of Adaptive Synthetic and Information Gain on C4.5 and Naive Bayes in Imbalance Class Dataset," International Journal of Advanced Science Computing and Engineering, vol. 4, no. 1, pp. 1–11, Jan. 2022, doi: 10.30630/IJASCE.4.1.70.

[29] U. Ahmed, J. C. W. Lin, and G. Srivastava, "Mitigating adversarial evasion attacks of ransomware using ensemble learning," Computers and Electrical Engineering, vol. 100, p. 107903, May 2022, doi: 10.1016/J.COMPELECENG.2022.107903

[30] S. A. EL_Rahman, R. A. AlRashed, D. N. AlZunaytan, N. J. AlHarbi, S. A. AlThubaiti, and M. K. AlHejeelan, "Chronic Diseases System Based on Machine Learning Techniques," International Journal of Data Science, vol. 1, no. 1, pp. 18–36, Apr. 2020, doi: 10.18517/IJODS.1.1.18-36.2020.

[31] N. M. Hatta, Z. A. Shah, and S. Kasim, "Evaluate the Performance of SVM Kernel Functions for Multiclass Cancer Classification," International Journal of Data Science, vol. 1, no. 1, pp. 37–41, May 2020, doi: 10.18517/IJODS.1.1.37-41.2020.