INTERNATIONAL CONFERENCE ON ADVANCES IN TECHNOLOGY AND COMPUTING (ICATC-2021)
Faculty of Computing and Technology (FCT), University of Kelaniya, Sri Lanka
18th December 2021

Index N0: CT-09-52

An Approach to Detect Fileless Malware that Maintains Persistence in Windows Environment

Malmi Atapattu (1st Author)
Department of Industrial Management
University of Kelaniya
Dalugama, Sri Lanka
malminatasha@gmail.com

Buddhika Jayawardena (2nd Author)
Department of Industrial Management
University of Kelaniya
Dalugama, Sri Lanka
buddhikaj@kln.ac.lk

***Abstract* — The rapid enhancement of the Internet in the past few years has increasingly impacted the general public's work and life. As a drawback, this enhancement has also led to a major increase in malicious software on the internet causing great security threats to the consumers of the internet. Currently, a new type of malware class called Fileless malware has come into action causing more destructive damages. As the name Fileless suggests, these types of malware programs are not files or executables, but a malicious activity that runs entirely in the memory, leaving the slightest evidence on the targeted host machine. Microsoft Windows is one of the most widely used operating systems both in personal desktop computers and enterprise computer systems and is highly targeted by Fileless malware. This paper provides an approach to detect fileless malware that maintains persistence in the Windows environment using Fileless malware behavioural data and deep learning-based classification models.**

Keywords — fileless malware, windows, deep learning

I. INTRODUCTION

It is predicted that the cost of global cybercrime will increase by 15 percent annually over the next five years which is estimated to reach $10.5 trillion USD annually by 2025. This projected cost includes theft of intellectual property, loss of personal and financial data, stolen money, restoration of destructed infrastructure and reputational harm [1]. The defensive systems, techniques and tools are improved continuously to avoid cybercrimes and to reduce the harm of consequences. But cybercrimes are consistently growing mainly due to the facts like an increase in the number of internet users, increase in the number of connected devices, increase in the variety and complexity of services, and increase of digitalization [2]. Almost every day technology consumers are introduced to new types of malware that cause them severe destructions including privacy-related issues. According to AV-TEST security reports, nearly 140 million new malware pieces have been introduced only during 2021.

Malware is basically a piece of software or a code that is inserted into a computer system with the objective of compromising the computer functions, stealing data, or evading access control without user knowledge. In order to evade detection, a malware uses different types of more advanced and sophisticated techniques. Thus, some variant of malware uses a technique that is running within the memory rather than from a file and those are called fileless malware [3]. For ease of reference, fileless malware can be categorized in various ways such as initially by their entry point, secondly by the form of the entry point, and finally by the type of the infected host [4]. A fileless malware is developed to be concealed from antivirus software making the detection difficult. The attack lifecycle of a fileless malware consists of various stages where stage 1 focuses on the initial delivery of attack, stage 2 targets the persistence, stage 3 is the execution and completes the attack once the objectives are met [5]. These stages are illustrated in Fig. 1.

Stage 1 Delivery	**Stage 2** Persistence	**Stage 3** Execution	**Achieve Objective**
• Exploits • Malicio-us Macro • Scripts etc	• WMI subscriptions • Registry • Task scheduler etc	• PowerShell • JavaScript • VBScript • Command line tools etc	

Fig. 1. Lifecycle of filelesss malware attack.

Stage 1. Delivery: Social engineering methods are often used by these attackers to commence the initial delivery. In this stage, two main strategies are used to evade detection by antivirus signature scanning which are, downloading directly into memory and use of trusted applications. Attackers tend to get a whitelisted and approved application; thus security software will not inspect legitimate software [5].

Stage 2. Persistence: To achieve persistence, attackers store the malicious code in unusual locations in the operating systems or common utilities such as the Windows registry, Windows Management Instrumentation (WMI) store, SQL tables or scheduled tasks. Code is injected into a system process and will seem like coming from a legitimate process [5].

Stage 3. Execution: After the persistence, malware will depend on Windows internals like PowerShell, JavaScript and Macro execution of the official documents and other legitimate executables to commence the execution [5].

Fileless malware attacks are initiated directly on most Windows applications and system administration tools like PowerShell and Windows Management Instrumentation (WMI) to exploit and spread the infection [6]. One of the main concerns behind Windows being a victim of fileless malware is the development of the Microsoft .NET framework. Though .NET was able to provide a significant contribution to enhance the software development process, unintentionally revolutionized the malware industry by providing a much easier surface for the malware coders to develop and spread malware and attain their unethical goals [5]. Windows administrative tools which are already installed on the victim's software can be easily used to launch the initial infection and also to commence the attack which is the main reason for the fileless malware to be a serious concern for Windows users [7].

According to WatchGuard's Internet Security Report for Q4 2020, fileless malware attack rates had grown by nearly 900% in 2020 compared to 2019 [8]. Due to the unforeseen fileless malware rising trends, many organizations are at a higher risk. Because of the fileless nature of these types of malware pieces, they are difficult to be detected by traditional anti-virus software. Windows OS is more vulnerable to fileless malware because of its system administration tools like PowerShell, WMI and also applications like Microsoft Office Macros. Though Microsoft has implemented next-gen anti-virus capabilities like Antimalware Scan Interface (AMSI), according to Sophos, AMSI is not a remedy for fileless malware since attackers are continuously finding methods to obfuscate malicious codes and bypass these kind of anti-virus solutions [9]. Therefore, same as other complex malware, fileless malware detection has been a serious concern for today's technological world.

Various traditional mechanisms [5], signature-based [10],[11] and heuristic-based techniques have been used to detect fileless malware. Due to the considerable drawbacks of signature-based techniques, some techniques have been implemented using heuristic-based machine learning and deep learning approaches. As machine learning models, classification algorithms like Perceptron [7], SVM, RF, XGB [12] and as deep learning models, MLP and CNN [13] have been mainly used. Most of the research works are limited to detecting PowerShell based malware and also there is a least deliberation towards malware persistence. Because of the freshness of this topic and the complexity of these types of malware, there is a lack of research regarding fileless malware detection and prevention techniques [7]. Thus, there is a noticeable gap in proper mechanisms to detect fileless malware that maintain persistence in the Windows environment.

The purpose of this research is to get a systematic approach to develop a solution that detects fileless malware which maintain persistence in the Windows environment. The research objectives are as follows: To identify the current techniques and mechanisms used by intruders to initiate fileless malware attacks on the Windows environment. Then, to identify the static and dynamic behaviours of fileless malware on Windows environment. Next, to develop a model to detect fileless malware that maintain persistence in the Windows environment using the identified behavioural data. Finally, to identify a suitable mechanism to test the validity of the proposed model.

In this study, a systematic approach is taken to address the concern of how to detect fileless malware that maintain persistence in the Windows environment. This approach includes the analysis of the current fileless malware techniques and the development of a mechanism to detect Windows fileless malware. The remainder of the paper is organized as follows: In section 2, a discussion is made on the review on the existing literature. Then in section 3, the methodology is described. Results and discussions are explained in section 4. Finally, in section 5, the conclusion is included with limitations and future work.

II. LITERATURE REVIEW

This study is an approach to investigate and understand the techniques that are being used to initiate fileless malware in the Windows environment by reviewing the existing literature. In addition, the identified techniques are mapped into an industry-standard framework called MITRE ATT&CK framework for further analysis.

In [5], fileless malware techniques are categorized based on the evasion techniques that are targeted by the malware authors which are malicious documents, malicious scripts, living off the land, and malicious code in memory. According to [14], fileless malware attack techniques are categorized into memory-resident malware, Windows registry malware, rootkits, process hollowing/injection attacks, reflective DLL injection, dynamic data exchange attacks, and dual-use tool attacks. Since this study mainly focuses on malware persistence, further discussion is made on persistence attack techniques.

Persistence attacks are mainly targeting to existing inside the victim's system for a longer period of time until the attack's goal is accomplished. To gain persistence, attackers use many evasion mechanisms such as malicious documents and scripts and living off the land techniques. Adversaries often compromise the legitimate Windows administration tools like PowerShell and WMI to evade detection and maintain persistence inside the system [15]. Further, these malware pieces are typically stored in unusual locations in the system like operating system utilities, WMI store, SQL tables, Windows registry, or OS task scheduler. While maintaining persistence, attackers look for the vulnerabilities in the system to exploit and steal data. Most of the time living in the system and running the fileless malware in the background as a service leads to a successful attack since the detection is difficult.

In Windows registry malware attacks, attackers mainly target to embed the malware deeply inside the Windows registry [14]. Windows registry is a system-level database that stores settings which are required for the operations of Windows OS and some applications. The malicious payload is injected into the registry and after accomplishing the malicious objective it can destruct itself without leaving traces. There are several attack methods in Windows registry malware. One is adding JavaScript code into the registry and

the code is executed when a legitimate application is running. Another method is process hollowing which is replacing the legitimate process in the memory with a malicious payload by injecting. More sophisticated attacks are capable of performing multiple process injections thus, the detection is entirely difficult.

In addition to the review on literature, these identified techniques were mapped with the MITRE ATT&CK framework which is a behavioural model and a curated knowledgebase created by the MITRE cooperation that includes information regarding cyber adversary behaviour signifying the different stages of an adversary's lifecycle and their targeted platforms. With this analysis, new possibilities of fileless malware techniques, tools, attack procedures are identified along with examples of the malware types that use the techniques. Table 1 defines the findings of the analysis.

Table 1. Techniques, tools, attack procedure and malware examples from ATT&CK framework

Technique	Tool	Attack Procedure	Malware Examples
Use of command and scripting interpreter	PowerShell	Abusing PowerShell commands and scripts for execution of malware, information discovery, download and run executables from the Internet.	APT19, APT28, APT29, PowerSploit, njRAT
	Windows Command Shell (cmd)	Batch scripting to automate the execution of the malware. Opening a reverse shell or a remote shell on the system to execute commands.	4H RAT, ABK, abdupd, admin@338, APT1, APT18, APT41
	JavaScript	Abusing various implementations of JavaScript to execute malicious behavior like hosting malicious scripts on websites or downloading and executing scripts as secondary payloads.	Poweliks, Astaroth, FIN6
Process Hollowing	API calls	Creating a process in a suspended state and hollowing its memory which then be replaced with malicious payload. Use of native Windows API calls like CreateProcess, ZxUnmapViewOfSection, NtUnmapViewOfSection, VirtualAllocEx	Agent Tesla, Astaroth, Bazar, Smoke Loader
Process Doppelganging	Windows Transactional NTFS (TxF)	Exploiting TxF to replace the memory of a legitimate process.	Bazar, Leafminer, SynAck
DLL/ PE Injection	PE files	Before loading the DLL, writing the path in the virtual address space of the target process.	Aria-body, ComRAT, PowerSploit
		Use of Windows native API calls such as VirtualAllocEx, WriteProcessMemory and CreateRemoteThread.	
Modify Registry	Command line utilities	Hiding configuration information within Registry keys. Gaining persistence in the system. Use of Reg command line utility for registry modification. Adding an entry to the "run keys" in the Registry hives or in startup folder.	ADVSTORES HELL, BADCALL, Cardinal RAT, Netwalker
Rootkits	Master Boot Record, System Firmware	Intercepting and modifying system API calls while living in the kernel level	ZeroAccess
Dynamic Data Exchange	Visual Basic for Applications (VBA) macros	Infecting MS Office applications with DDE commands	APT28, APT37

III. METHODOLOGY

This section describes the methodology utilized to perform the research, including the behaviour analysis of fileless malware using a sandbox environment and data collection. Finally, to determine a suitable deep learning based fileless malware classification model.

A. Fileless Malware Behavior Analysis

In this section further discussion is made on the behaviour of fileless malware using the behaviour reports of Poweliks malware from the Cuckoo sandbox.

Cuckoo sandbox is an open-source automated malware analysis sandbox [16], which consists of features like tracing API calls and general behavior of the files, and advanced memory analysis [17]. When creating the sandbox environment for the analysis, Cuckoo sandbox was chosen since its advanced features and functionalities. For the host environment a PC with 8GB memory, 1TB hard disk and 4 core processer was used. Ubuntu LTS 18.04 was installed as the operating system for the host environment. The guest environment was created using Virtual Box 5.2.4 with hardware specifications of 2GB memory and 32GB hard disk. Windows 7 operating system was used for the virtual machine and Google Chrome, MS Office applications, and Adobe Reader was installed as an additional software.

Poweliks malware is one of the first fileless malware that has the ability to do a Ransomware infection on a computer system [18]. According to Symantec report [19] on Poweliks malware was an evolution of the file-based malware called Wowliks. This malware uses registry manipulation and persistence techniques during the attack. When the JavaScript script which contains the malicious payload runs, a new registry entry will be added to the registry. This is then used

for the fileless execution. Using an alternate data stream, the original file is deleted leaving no traces in the system but the registries will be maintaining the persistence continuously contacting the malicious command and control server for information stealing or other purposes.

A sample Poweliks trojan from 'VirusShare' was executed on the Cuckoo sandbox environment for behaviour analysis. Using the JavaScript command, it had created a registry key to a long series of bytes to store the malicious configurations. As a persistence technique, it had installed itself for autorun at Windows startup which is shown in Fig. 2.

reg_key	HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run\	reg_value	rundll32.exe javascript:"\..\mshtml, RunHTMLApplication ";document.write("\74script language=jscript. encode>"+(new ActiveXObject("WScript. Shell")). RegRead("HKCU\\ software\\microsoft\\windows\\currentversion\\run\\\")+"\74/ script>")

Fig. 2. Installation at Windows Startup for autorun.

Moreover, it allocated execute permission to another process and resumed a suspended thread in a remote process which is potentially indicative of possible code injection. Windows command-line utility had used for the command execution and other applications were not used during the execution. Further, this malware had tried to reach some IP addresses rather than a domain which did not respond back that can be potentially indicative of command-and-control traffic. To conclude, this behaviour of Poweliks malware which is a real-world example of fileless malware indicates previously discussed evasion techniques.

B. Data Collection

'VirusShare' malware repository and 'theZoo' malware repository was used to get the malware samples. The selected malware sample includes malicious Windows PE files, .NET binaries, known Windows fileless malware, PowerShell and WMI based malware. To create the dataset, initially, each malware was executed in Cuckoo sandbox environment and collected the reports. For this purpose, the same environment that was used for the behavior analysis was used with same conditions. Each collected report is in JSON format and contains all the information regarding the execution process including API calls, memory buffer and string values. All together 3085 malware sandbox reports were collected. Benign malware samples were also collected from Portable Freeware collection [20]. The collected sample also executed in the same sandbox environment and 1100 benign malware reports were collected.

C. Data Preprocessing

For data preprocessing, API calls were selected from registry and system categories because according to the identified fileless malware techniques system and registry related API sequence will be effective when determining fileless persistence attacks. API call sequences for each malware was converted to the given unique index value of the sandbox report. The same technique was used for the benign samples as well. After the sequential data, in the last column of the CSV file '1' was given if it was malware and '0' was given if it was benign malware. The prepossessed data was then used to train the sequence data classification model.

D. Sequence Data Classification

API call sequence represents the behaviour of the malware across certain period. This behavioural data can be trained using a classification algorithm to develop a model that detects fileless malware from its behaviour. For this purpose, deep learning classification was chosen because, as specified by the authors, there is a higher detection ability for deep learning when compared to traditional shallow learners such as SVM [21]. For this paper, two deep learning-based sequential classification models that are LSTM and BI-LSTM models were used to train the dataset.

1) Long short-term memory (LSTM): LSTM is a variant of the recurrent neural network (RNN) which was proposed by Hochreiter and Schmidhuber who had applied three gates to solve the vanishing gradient problem of RNN [22].

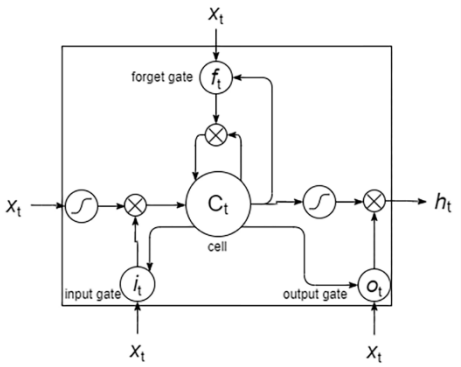

Fig. 3. A Long Short-Term Memory Cell.

LSTM networks are similar to RNNs except for the updates of the hidden layer are replaced by purpose-built memory cells. Therefore, LSTM provides better results for a long range of data with dependencies. The structure of the LSTM cell [23] is represented in Fig. 3 and it can be implemented as follows:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \quad (1)$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \quad (2)$$

$$c_t = f_t c_{t-1} + i_t tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \quad (3)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_{t-1} + b_o) \quad (4)$$

$$h_t = o_t tanh(c_t) \quad (5)$$

In Equation (1), (2), (3), (4), (5), where □ is the logistic sigmoid function, and i, f, o, c, are input gate, forget gate, output gate and cell vectors respectively. The cell vectors and hidden vector h, all are in the same size. W is the weight matrix, and b is the bias vector. The activation value h_t of the hidden unit at time step t can be calculated using the information at various times [23]. It is possible because the gating mechanism works through storing the historical memory.

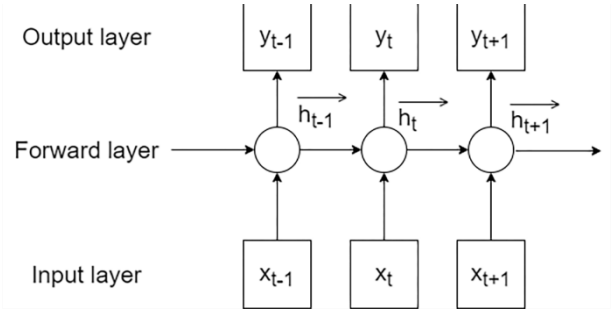

Fig. 4. A Long Short-Term Memory Model.

2) Bi-directional LSTM (BI-LSTM): BI-LSTM inputs run in two ways, that are in forward and backward directions. Therefore, forward states and backward states for a specific time frame can be utilized efficiently.

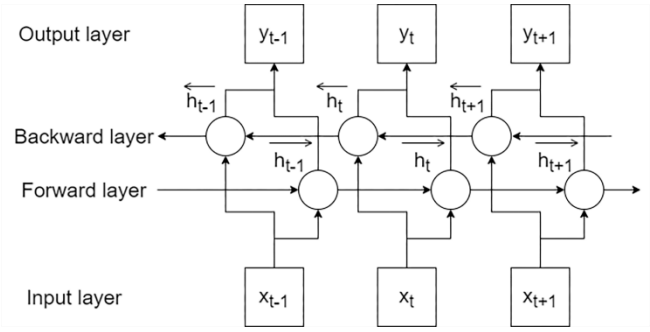

Fig. 5. A Bi-directional Long Short-Term Memory Model.

BI-LSTM connects the two hidden layers to the same output layer which is shown in Fig. 5 [24].

E. Model Trainig

For the model training, preprocessed dataset was used. The dataset includes 3050 malware API sequences and 1050 benign malware API sequences. The dataset was split to approximately 6:1 ratio for training and testing respectively.

Table 2. Dataset information

	Training	**Testing**	**Total**
Data (malware & benign)	3500	600	4100

Both LSTM and BI-LSTM models were trained using the dataset. In Table 3 it shows the parameters that were used in model training.

Table 3. Parameter table

Training Parameters	**LSTM**	**BI-LSTM**
Embedding	Word2vec	Word2vec
Batch size	64	256
Steps per Epoch	250	300
Optimizer	Adam	Adam
Activation Function	Sigmoid	Sigmoid
Loss Function	Binary cross entropy	Binary cross entropy

IV. RESULTS AND DISCUSSION

This section includes the findings of the research and performance evaluation. The sequence labeled dataset was trained using LSTM and BI-LSTM algorithms to select the suitable model for the fileless malware detection.

Table 4. Confusion Matrix

Reality / Predicted	**Malware**	**Benign**
Malware	TP	FP
Benign	FN	TN

Based on the confusion matrix for binary classification that is shown in Table 4, *Accuracy* of each model can be obtained (6).

$$Accuracy = TP + TN / (TP + FP + FN + TN) \quad (6)$$

Tensorflow and Keras libraries have been used to construct, train, and evaluate the fileless malware and benign API sequence classification.

A. LSTM Performance

For LSTM the highest accuracy received is 0.88 when using 3 LSTM layers. Fig. 6 shows the variation of training accuracy and average validation accuracy of fileless malware API sequence classification.

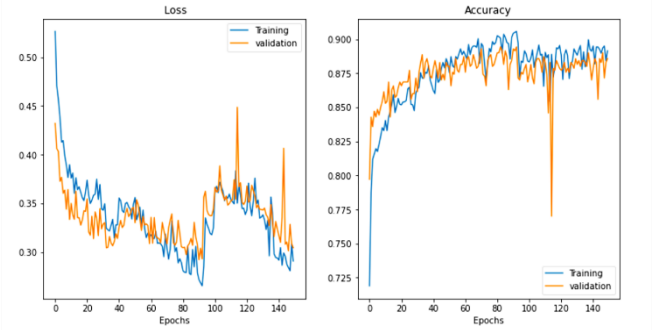

Fig. 6. Training and validation accuracy, average training and validation loss comparison of LSTM classification.

B. BI-LSTM

For BI-LSTM highest accuracy received is 0.92 when using 3 layered BI-LSTM model. Fig. 7 shows the variation of training accuracy and average validation accuracy of fileless malware API sequence classification using BI-LSTM.

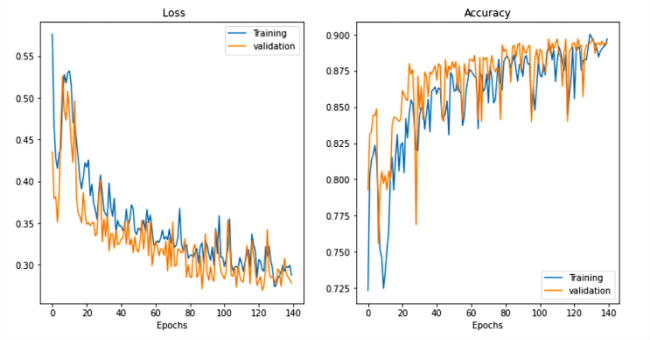

Fig. 7. Training and validation accuracy, average training and validation loss comparison of BI-LSTM classification.

From the above comparison it is clearly visible that BI-LSTM shows a higher accuracy in detecting fileless malware API sequences rather than LSTM model.

Proposed fileless malware detection mechanism is compared with a similar study that uses fileless malware API sequences in a dynamic signature model [10]. As a major drawback, it is stated that there is a higher false positive rate. Moreover, the database should be updated regularly with signatures of API call sequences which is difficult because of the current rapid evolvement of fileless malware. Therefore, our model is more applicable for detecting fileless malware using the API sequences.

V. CONCLUSION

In this paper, we presented a model for detecting fileless malware that maintains persistence in the Windows environment. For this purpose, initially, we identified the techniques and mechanisms used by intruders to initiate fileless malware using the existing literature and mapping the findings with the MITRE ATT&CK framework. According to the findings we identified that malware API sequence would be suitable for detecting fileless malware that maintains persistence. After extracting fileless malware API sequence data from sandbox reports, we trained two models which are LSTM and BI-LSTM to determine the most suitable model. Based on the highest accuracy we concluded that the BI-LSTM model performs efficiently in detecting fileless malware API sequences.

As future work, improvements will be made on the model by combining with other variants of deep learning classification methods.

REFERENCES

[1] S. Morgan, “Cybercrime To Cost The World $10.5 Trillion Annually By 2025,” 2020. https://cybersecurityventures.com/cybercrime-damages-6-trillion-by-2021/ (accessed Aug. 20, 2021).

[2] L. Caviglione *et al.*, “Tight Arms Race: Overview of Current Malware Threats and Trends in Their Detection,” *IEEE Access*, vol. 9, pp. 5371–5396, 2021.

[3] A. Alshamrani, S. Myneni, A. Chowdhary, and D. Huang, “A Survey on Advanced Persistent Threats: Techniques, Solutions, Challenges, and Research Opportunities,” *IEEE Commun. Surv. Tutorials*, vol. 21, no. 2, pp. 1851–1877, 2019.

[4] “Fileless threats.” https://docs.microsoft.com/en-us/windows/security/threat-protection/intelligence/fileless-threats (accessed May 06, 2021).

[5] B. N. Sanjay, D. C. Rakshith, and V. H. Vinay, “An Approach to Detect Fileless Malware and Defend its Evasive mechanisms,” *2018 3rd Int. Conf. Comput. Syst. Inf. Technol. Sustain. Solut.*, pp. 234–239, 2018.

[6] V. Khushali, “A Review on Fileless Malware Analysis Techniques,” vol. 9, no. 05, pp. 46–49, 2020.

[7] A. G. Bucevschi, G. Balan, and D. B. Prelipcean, “Preventing File-Less Attacks with Machine Learning Techniques,” *Proc. - 21st Int. Symp. Symb. Numer. Algorithms Sci. Comput. SYNASC 2019*, pp. 248–252, 2019.

[8] “New Research: Fileless Malware Attacks Surge by 900% and Cryptominers Make a Comeback, While Ransomware Attacks Decline,” 2021. https://www.watchguard.com/wgrd-about/press-releases/new-research-fileless-malware-attacks-surge-900-and-cryptominers-make (accessed Jun. 25, 2021).

[9] C. Osborne, “This is how attackers bypass Microsoft’s AMSI anti-malware scanning protection,” 2021. https://www.zdnet.com/article/this-is-how-attackers-bypass-icrosoft-antimalware-scan-software-amsi/ (accessed May 20, 2021).

[10] R. Tarek, S. Chaimae, and C. Habiba, “Runtime API Signature for Fileless Malware,” *Adv. Inf. Commun. Proc. 2020 Futur. Inf. Commun. Conf.*, vol. 1, no. AISC 1129, pp. 645–654, 2020.

[11] N. R. Mistry and M. S. Dahiya, “Signature based volatile memory forensics: a detection based approach for analyzing sophisticated cyber attacks,” *Int. J. Inf. Technol.*, vol. 11, no. 3, pp. 583–589, 2019.

[12] M. Mimura and Y. Tajiri, “Static detection of malicious PowerShell based on word embeddings,” *Internet of Things*, vol. 15, p. 100404, 2021.

[13] X. Huang, L. Ma, W. Yang, and Y. Zhong, “A Method for Windows Malware Detection Based on Deep Learning,” *J. Signal Process. Syst.*, vol. 93, no. 2–3, pp. 265–273, 2021.

[14] A. Afreen, M. Aslam, and S. Ahmed, “Analysis of Fileless Malware and its Evasive Behavior,” *1st Annu. Int. Conf. Cyber Warf. Secur. ICCWS 2020 - Proc.*, 2020.

[15] S. Quintero-Bonilla and A. M. del Rey, “A new proposal on the advanced persistent threat: A survey,” *Appl. Sci.*, vol. 10, no. 11, 2020.

[16] Heena, “Advances In Malware Detection- An Overview,” 2021, [Online]. Available: http://arxiv.org/abs/2104.01835.

[17] “Cuckoo.” https://cuckoosandbox.org/ (accessed Aug. 20, 2021).

[18] B. Alsulami and S. Mancoridis, “Behavioral Malware Classification using Convolutional Recurrent Neural Networks,” *MALWARE 2018 - Proc. 2018 13th Int. Conf. Malicious Unwanted Softw.*, pp. 103–111, 2019.

[19] L. O’murchu and F. P. Gutierrez, “SECURITY RESPONSE The evolution of the fileless click-fraud malware Poweliks,” 2015. [Online]. Available: https://autoblog.postblue.info/autoblogs/lamaredugoffrblog_a1de86d064e376dc283723997fd86bde6ba2d492/media/456c7abf.evolution-of-poweliks.pdf.

[20] “Portable Freeware.” https://www.portablefreeware.com/ (accessed Oct. 10, 2021).

[21] D. Javaheri, P. Lalbakhsh, and M. Hosseinzadeh, “A Novel Method for Detecting Future Generations of Targeted and Metamorphic Malware Based on Genetic Algorithm,” *IEEE Access*, vol. 9, pp. 69951–69970, 2021.

[22] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Comput.*, vol. 9, pp. 1735–1780, 1997.

[23] Y.-W. Lu, C.-Y. Hsu, and K.-C. Huang, “An Autoencoder Gated Recurrent Unit for Remaining Useful Life Prediction,” *Processes*, vol. 8, no. 1155, 2020.

[24] Z. Huang, W. Xu, and K. Yu, “Bidirectional LSTM-CRF Models for Sequence Tagging,” 2015, [Online]. Available: http://arxiv.org/abs/1508.01991.