Machine Learning Approaches to Malicious PowerShell Scripts Detection and Feature Combination Analysis

Hsiang-Hua Hung, Jiann-Liang Chen, Yi-Wei Ma[*]

Department of Electrical Engineering, National Taiwan University of Science and Technology, Taiwan
m10907506@gapps.ntust.edu.tw, Lchen@mail.ntust.edu.tw, ywma@mail.ntust.edu.tw

Abstract

With advances in communication technology, modern society relies more than ever on the Internet and various user-friendly digital tools. It provides access to and enables the manipulation of files, trips, and the Windows API. Attackers frequently use various obfuscation techniques PowerShell scripts to avoid detection by anti-virus software. Their doing so can significantly reduce the readability of the script. This work statically analyzes PowerShell scripts. Thirty-three features that were based on the script's keywords, format, and string combinations were used herein to determine the behavioral intent of the script. Ones are characteristic-based features that are obtained by calculation; the others are behavior-based features that determine the execution function of behavior using keywords and instructions. Behavior-based features can be divided into positive behavior-based features, neutral behavior-based features, and negative behavior-based features. These three types of features are enhanced by observing samples and adding keywords. The other type of characteristic-based feature is introduced into the formula from other studies in this work. The XGBoost model was used to evaluate the importance of the features that are proposed in this study and to identify the combination of features that contributed most to the detection of PowerShell scripts. The final model with the combined features is found to exhibit the best performance. The model has 99.27% accuracy when applied to the validation dataset. The results clearly indicate that the proposed malicious PowerShell script detection model outperforms previously developed models.

Keywords: Machine learning, XGBoost, PowerShell, Malicious scripts, Behavioral features analysis

1 Introduction

PowerShell provides various functions for connecting and manipulating files, programs, and Windows APIs [1]. It can help system administrators rapidly operate management systems and achieve automation. It is the tool of choice for many attackers due to its flexibility, robust structure, and ability to execute scripts directly from the command line. Attackers have also developed various methods to obfuscate PowerShell scripts to avoid detection by anti-virus software and have even developed automated tools to obfuscate scripts, such as code tags, characters, and abstract syntax trees [2].

Since 2016, Fileless Malware, a new attack technique, has been attracting much attention in the limelight [3]. While system administrators widely use PowerShell to manage their computers, attackers frequently use this framework to conduct attacks [4]. It has become one of the mainstream techniques for attacking native applications [5]. In 2018, a significant increase in the number of attacks of this type was observed, as attackers used native applications on computers to map malicious code directly into memory to attack without writing to the disk, including attacks via PowerShell [6]. Dual-use PowerShell tools accounted for the largest share, 23%, of threats that were detected on endpoints in the second half of 2020 [7]. Because of the global impact of COVID-19, attackers used the theme of the epidemic to motivate victims to click on emails and exploit the vulnerability of workers at home and telecommuting to conduct cyber attacks, leading to the growth of the Donoff Malware threat. The Q3 2021 Trellix ATR Report found that [8].

This study proposed machine learning approaches to malicious PowerShell script detection and feature combination analysis has the following contributions. (1) This study uses static analysis to detect PowerShell scripts and determine their behavioral intent based on their keywords, formats, and string combinations. (2) Two feature combinations are proposed; they are characteristic-based and behavior-based. Behavior-based feature combinations can be divided into positive behavior-based, neutral behavior-based, and negative behavior-based. (3) A feature used in a previous study was added to the characteristic-based feature combination. (4) The three features are enhanced by observing samples and adding keywords, and performance analysis was conducted to ensure that the features that are proposed in this study are helpful in the more effective identification of malicious scripts. (5) A total of 33 features support the high performance of the XGBoost algorithm.

This study uses static analysis to detect PowerShell scripts and determine their behavioral intent based on their keywords, formats, and string combinations. Two types of feature combinations are proposed; they are characteristic-based and behavior-based. Behavior-based feature combinations can be divided into positive behavior-based, neutral behavior-based, and negative behavior-based. A feature that was used in a previous study was added to characteristic-based of feature combination. The three

*Corresponding Author: Yi-Wei Ma; E-mail: ywma@mail.ntust.edu.tw

DOI: 10.53106/160792642024012501014

features are enhanced by observing samples and adding keywords, and a performance analysis was conducted to ensure that the features that are proposed in this study are helpful in the more effective identification of malicious scripts. A total of 33 features support the high performance of the XGBoost algorithm.

2 Related Works

2.1 PowerShell Malware Threats

This section reviews the current status of the PowerShell malware threat. Kumar et al. [9] investigated and listed research questions on new fileless malware. Afreen et al. [10] observed a shift in information security threats. Afreen et al. concluded by presenting the critical elements of the AVT response strategies and techniques, such as behavioral analytics, logging, least privilege rule, and content filtering.

2.2 De-obfuscation Techniques

Liu et al. [11] performed the detection and anti-obfuscation for malicious PowerShell malware. They proposed a method for the anti-obfuscation and analysis of malicious PowerShell scripts that are embedded in Word files. Ugarte et al. [12] introduced a static and dynamic multi-level de-obfuscator for PowerShell attacks that was named PowerDrive. They also provided a taxonomy of behavioral models that are used to analyze the code and a complete list of malicious domains that are contacted during the analysis.

2.3 Detection of Malicious PowerShell Scripts

In recent years, PowerShell de-obfuscation techniques have matured, and many studies have focused on using artificial intelligence to detect and analyze PowerShell scripts. Hendler et al. [13] analyzed PowerShell malicious scripts using a neural network detector.

Rusak et al. [14] proposed a hybrid approach that combines traditional program analysis with abstract syntactic trees and deep learning. Li et al. [15] designed a semantics-aware PowerShell attack detection system to identify 31 new semantic signatures using classical target-oriented association mining algorithms for PowerShell attacks. Tajiri et al. [16] construct word-level language models.

Fang et al. [17] proposed a hybrid feature-based model. The model analyzes malicious and benign scripts. They also used a word embedding and text classification model, FastText, to extract semantic features and to detect automatically malicious PowerShell scripts. They emphasized that a mixture of manual and automatic features can effectively enhance the performance of a detection model.

Song et al. [18] proposed an AI-based approach to feature optimization to improve the accuracy of detection of malicious PowerShell scripts. The optimized features are trained in models of machine learning and deep learning. Choi [19] proposed a GCN-based approach to detecting malicious PowerShell scripts by extracting feature data from previously identified PowerShell scripts and calculating the Jaccard similarity between the new PowerShell and the existing PowerShell scripts. Alahmadi et al. [20] presented the MPSAutodetect model for the automatic detection of malicious PowerShell scripts using stacked denoising auto-encoders (SdAs). Their model extracts meaningful features and feeds the valuable ones into the XGBoost classifier. The main feature of MPSAutodetect is that the model does not require the manual extraction of features, eliminating the need for the manual finding of features.

3 Proposes System

The system architecture consists of five parts which are data collection, data processing, feature definition, model training, and model prediction. Figure 1. shows the system. The proposes system includes data collection module, data processing module, feature definition module, model training module and model prediction module. In the data collection phase, the malicious script uses psencmds, which is a public dataset that was provided by the Unit 42 team of Palo Alto Networks [21]. The benign script for this work is a crawler for obtaining PowerShell-type scripts on the GitHub platform, and the two are mixed in datasets of similar numbers of benign and malicious scripts. In the data processing stage, the content of the scripts is tagged, and incomplete data are filtered out.

In the feature definition stage, 33 features are extracted and categorized as characteristic-based or behavior-based, based on observations of the results of labeling after data processing, for subsequent training of the model used in this work. In the model training and prediction stages, the XGBoost model is used herein to evaluate the importance of various combinations of features. In order to ensure the stability of the model and prevent overlearning, it was validated during the training process. The final results of this system in the detection of benign and malicious scripts are presented.

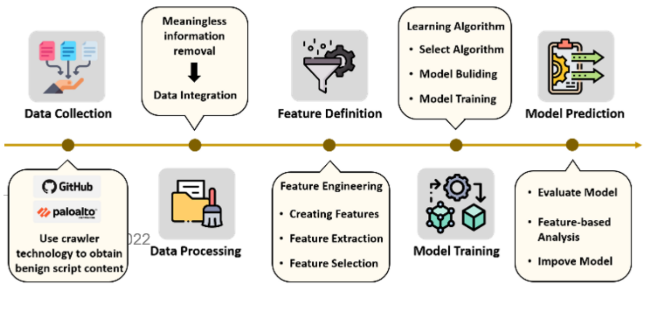

Figure 1. Proposed system architecture

3.1 Data Collection

The section introduces basic information about datasets that are used in this work. First, the main techniques used to obtain the data are described. The datasets that are used in this study have two kinds of sources. The first kind is open-source code platforms from which are obtained a large number of random scripts that are identified and verified as benign. The other source is a well-known information security team, which provided samples of malicious attacks.

The benign scripts are collected from GitHub, using crawling techniques to collect open-source PowerShell-type scripts on its platform. Validation that the samples collected are benign is performed through two malware detection and analysis services, Windows Defender and Virustotal. A total of 5,189 scripts were collected.

The malicious scripts were obtained using psencmds, a public dataset that is provided by the Palo Alto Networks Unit 42 team, using AutoFocus, which is the Palo Alto Networks' highly realistic threat intelligence service. A total of 4,100 PowerShell-type samples contained both command and script-type content. These samples reveal which techniques are widely used in PowerShell attacks.

3.2 Data Preprocessing

In the experiment, incomplete scripts and samples that cannot be adequately run with PowerShell are discarded to ensure that all data are available for subsequent tagging. The source of the malicious scripts that are used in this study was the Unit 42 team, who obtained them in 2017 using AutoFocus, which is a high-fidelity threat intelligence service. The benign scripts were obtained on GitHub using crawler technology. To ensure the integrity of the dataset by assimilating the data from different sources, we followed the Unit 42 team's behavior tagging approach for PowerShell scripts to merge benign and malicious datasets.

The number of datasets that were used in this study is 9289. Pre-processing to filter out invalid samples left a total of 8197 datasets. The datasets are split into 80%: 20% by Scikit-learn. 80% of datasets are used for training, and 20% are used for model validation.

3.3 Feature Definition

Table 1 lists the main contributions of this study. A total of 33 features are divided into two types, which are characteristic-based and behavior-based. Behavior-based features can be subdivided into positive, neutral, and negative behavior-based. The algorithm that is used in machine learning focuses depends on the weighting of the feature data, so feature extraction is crucial. In addition, the proposed and reinforced features are marked in red in the table.

Table 1. Thirty-three features

Feature Set: 33 Features					
F1	*One Liner*	F12	*Script Execution*	F23	*Byte Usage*
F2	*Variable Extension*	F13	*Crypto*	F24	*Negative Content*
F3	*Abnormal Size*	F14	*Enumeration*	F25	*Known Malware*
F4	*Obfuscation*	F15	*Hidden Window*	F26	*Code Injection*
F5	*Information Entropy*	F16	*Custom Web Field*	F27	*DNS C2*
F6	*Positive Content*	F17	*Persistence*	F28	*AppLocker Bypass*
F7	*Script Logging*	F18	*Registry*	F29	*AMSI Bypass*
F8	*FunctionBody*	F19	*Sleeps*	F30	*Embedded File*
F9	*License*	F20	*SysInternals*	F31	*Clear Logs*
F10	*Downloader*	F21	*Compression*	F32	*Disabled Protections*
F11	*Start Process*	F22	*Uninstalls Apps*	F33	*Screenshot*

3.3.1 Characteristic-based Features

Characteristic-based features are calculated or described in a state based on the content presented in the PowerShell script. F1 to F3 are intuitive features that observe or calculate states to measure and mark the script. F4 and F5 are the results that are obtained by calculating various attributes, the simultaneous annotation of which requires a large amount of data and allows the machine learning model to perform grouping and generalization. Table 2 presents the characteristic-based features.

F5 Information Entropy is the average amount of information content of all words in a text. Obfuscated scripts are to make it challenging to understand the intention and content. According to Choi et al. [22], the presence of obfuscated characters reduces entropy. Fang et al. [17] also applied a method to calculate the information entropy from PowerShell scripts.

Table 2. Characteristic-based features

ID	Name	Description
F1	One Liner	The script that solely exists on one line.
F2	Variable Extension	The wildcard «*» is used when setting the variable in the script.
F3	Abnormal Size	The size of the script is abnormal, the word count is too large or the line count is too high.
F4	Obfuscation	Calculate various properties of obfuscated scripts and observe the unique properties of malicious obfuscated scripts.
F5	Information Entropy	The average amount of information contained in each script received.

Table 3. Positive-behavior features

ID	Name	Description
F6	Positive Content	Keywords with positive meaning, such as: the official development package name.
F7	Script Logging	Use commands related to logging.
F8	FunctionBody	Use Annotated Functions in the script.
F9	License	Annotate copyright notices in the script.

3.3.2 Positive Behavior-based Features

Behavior-based type of features are used to observe the function or keyword of each command in the script, and they are classified into three types. Positive behavior-based means that the command or keyword has a positive meaning, indicating that the script has used the command correctly. F6 is a keyword that is observed to have positive behavioral meaning, and F7 to F9 are instructions or syntax that are observed to have positive behavioral meanings. Table 3 presents positive behavior-based features.

F6 Positive Content mainly refers to management-type scripts that commonly perform management functions on enterprise endpoints. Such scripts will appear in the command or keyword, such as ToolTip or Readme. We have enhanced the feature of F6 with new keywords for the MessageBox and WinGet functions. MessageBox means that the script has designed the user interaction function with the purpose of building a GUI interface; WinGet implies that the script has used the official package management module.

3.3.3 Neutral Behavior-based Features

Behavior-based type of features are used to observe the function or keyword of each command in the script, and they are classified into three types. Neutral behavior-based features are those the instruction or keyword is in a neutral position and may exist in both malicious scripts and benign scripts. Table 4 presents neutral behavior-based features.

Table 4. Neutral-based features

ID	Name	Description
F10	Downloader	Commands to download files locally or load files into memory.
F11	Start Process	Commands to start one or more processes on the local computer.
F12	Script Execution	Commands to execute commands or expressions on the local computer.
F13	Crypto	Commands to create symmetric and asymmetric encryption sub-objects.
F14	Enumeration	Commands to search for shared resources and user information in the system.
F15	Hidden Window	Commands to set the window of the session to be hidden.
F16	Custom Web Field	Commands to insert the new header and its value into the HttpHeaders collection.
F17	Persistence	Commands to set the process using schtacks or windows service.
F18	Registry	Commands to working with Registry Keys.
F19	Sleeps	Commands to suspends the activity in a script or session for the specified period of time.
F20	SysInternals	Commands to use Windows Sysinternals tools host advanced system utilities and technical information. ex: ProcDump.
F21	Compression	Commands to compress or decompress data.
F22	Uninstalls Apps	Commands to uninstall the app.
F23	Byte Usage	Commands to writes the specified byte array to the file.

Table 5. Negative behavior-based features

ID	Name	Description
F24	Negative Content	Keywords with negative meaning, commands frequently used in malicious scripts.
F25	Known Malware	Regnex patterns or collections of keywords that uniquely identify known malicious scripts.
F26	Code Injection	A combination of commands and keywords means code injection software attacks.
F27	DNS C2	A combination of commands and keywords means use DNS to gain control of C2 (Command and Control).
F28	AppLocker Bypass	A combination of commands and keywords means bypassing some warnings and tools using the regsvr32 registry.
F29	AMSI Bypass	A combination of commands and keywords means bypassing anti-malware scanning tool AMSI.
F30	Embedded File	Keywords for embedding in DOS MZ executables.
F31	Clear Logs	Commands to clears event log records executed by scripts.
F32	Disabled Protections	Commands to disable protection programs such as anti-spyware programs, malware scanners.
F33	Screenshot	Commands to execute the action of the screenshot.

3.3.4 Negative Behavior-based Features

Behavior-based type of features are used to observe the function or keyword of each command in the script, and they are classified into three types. Negative behavior-based refers to an instruction or keyword that has a negative meaning or is characterized by aggressive intent in their execution. Table 5 presents negative behavior-based features.

F24 Negative Content refers to a keyword that is commonly found in malicious scripts but cannot be classified by function. Attackers often use text storage sites such as Pastebin to download actual malware. Some red teams release scripts for penetration testing on the network, and attackers may intercept or use fragments of these scripts directly. This feature compares the behavior of scripts to determine whether it is used in penetration testing scripts. Two additional security analysis scripts are added to strengthen this feature.

F25 Known Malware compares scripts with known malicious scripts such as PowerSploit, nishang, and others, to determine if the script of interest includes a known malicious script. The new known malicious script PowerMemory is added to strengthen this feature.

3.4 Architecture of Detection Model

The two standard algorithms in ensemble learning are: bagging and boosting. The XGBoost classification model, based on the boosting algorithm, is used in this study [23]. Although the training speed and memory consumption are not comparable to those of LightGBM, the overall accuracy of XGBoost is superior, and the model is well developed. XGBoost has excellent parameter tuning and is less prone than LightGBM to overfitting.

The experiment that focuses on the classification of benign and malicious scripts is performed. The boosting calculation is performed using a tree-based model that is called gbtree. The maximum depth of the tree is initially set to a default value of six. The optimal number of trees in the XGBoost is 100.

4 Performance Analysis

The XGBoost model was used to obtain the feature importance scores of each feature type to evaluate the ability of the model to detect malicious PowerShell scripts.

4.1 Analysis of Performance in Strengthening Features

This section presents feature-based analysis and experiments using the XGBoost model. Its purpose is to analyze the main feature items that affect the model and evaluate the performance thereof. Finally, results are presented for a mixture of various feature types.

4.1.1 Analysis of Characteristic-based Features

F1 to F5 are characteristic-based features. Figure 2 show the results of performance analysis by XGBoost. Information Entropy (F5) is the most robust feature, followed by OneLiner (F1) and Obfuscation (F4). Therefore, the value of the new Information Entropy feature in this work has differs between benign and malicious scripts. The accuracy, precision, recall, F1 score, log loss and AUC are 77.38%, 100%, 55.19%, 71.13%, 7.8133 and 77.60% before strengthening. The accuracy, precision, recall, F1 score, log loss and AUC are 81.04%, 100%, 61.46%, 76.13%, 6.5497 and 80.73% after strengthening.

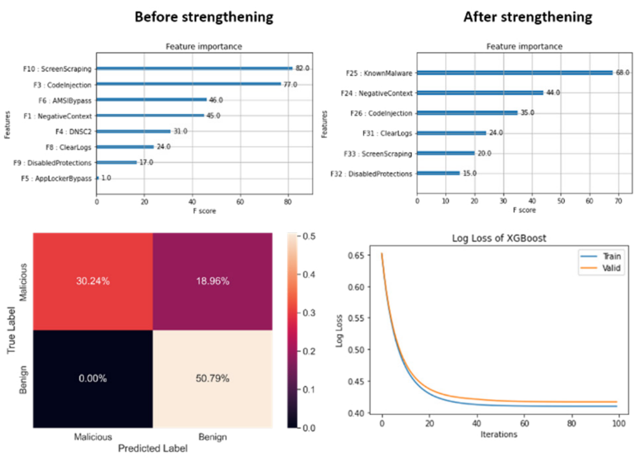

Figure 2. Performance of characteristic-based features

4.1.2 Analysis of Positive Behavior-based Features

F6 to F9 are positive behavior-based features. Figures 3 present the results of the feature importance analysis by XGBoost. The Positive Content (F6) was the worst feature. The experimental results showed that the enhanced Positive Content (F6) improved the detection effect. The accuracy, precision, recall, F1 score, log loss and AUC are 98.48%, 98.68%, 98.33%, 98.50%, 0.5265 and 98.48% before strengthening. The accuracy, precision, recall, F1 score, log loss and AUC are 99.09%, 99.04%, 99.16%, 99.10%, 0.3159 and 99.08% after strengthening.

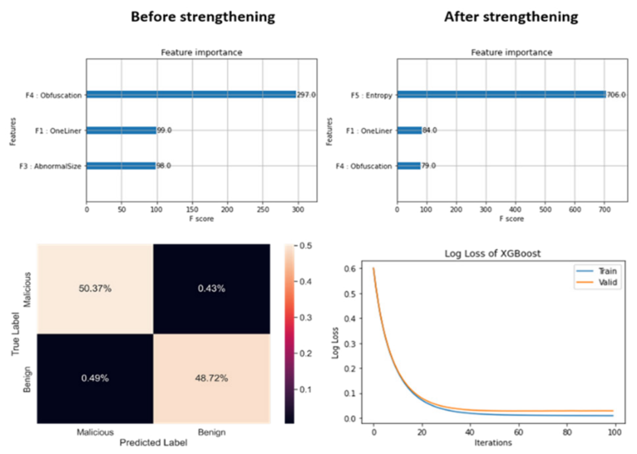

Figure 3. Performance of positive behavior-based features

4.1.3 Analysis of Negative Behavior-based Features

F24 to F33 are negative behavior-based features. Figure 4 show the results of the feature importance analysis by XGBoost. The enhanced Negative Content (F24) and Known Malware (F25) exhibited improved effectiveness. Known Malware (F25) identifies known malware, and Negative Content (F25) enhances the keywords in the red team exercise scripts. The accuracy, precision, recall, F1 score, log loss and AUC are 99.06%, 83.83%, 99.52%, 91.00%, 3.4328 and 89.97% before strengthening. The accuracy, precision, recall, F1 score, log loss and AUC are 92.07%, 86.73%, 99.64%, 97.74%, 2.7378 and 91.95% after strengthening.

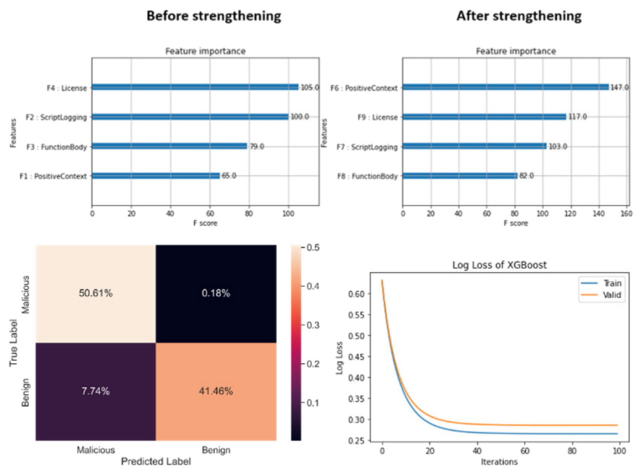

Figure 4. Performance of negative behavior-based features

4.2 Comparison of Study

Table 6 compares the predictive performances of models that are proposed in this and previous studies that are based on the same dataset. The XGBoost algorithm is applied to the same dataset for training and testing. The malicious scripts in this and the comparative studies are based on the public dataset psencmds, and the benign scripts are extracted from the GitHub platform.

Table 6. Comparison with the results of different studies

	Effective method for detecting malicious PowerShell scripts based on hybrid features (2021)	Our proposed system
Dataset	GitHub + Psencmds	
Algorithm	Random Forest	XGBoost
Accuracy	Original scripts: 98.93% Mixed scripts: 97.76%	99.27%
Precision	97.79%	99.52%
Recall	97.69%	99.04%
F1 Score	97.73%	99.28%

The proposed detection model outperforms other models with respect to accuracy by 0.3%, precision by

1.7%, recall by 1.3%, and F1 score by 1.5%. Therefore, the proposed detection model, based on the XGBoost algorithm, excellently classifies benign and malicious PowerShell scripts.

4.3 Summary

In this study, 33 features were divided into five groups, which are characteristic-based, positive behavior-based, neutral behavior-based, negative behavior-based, and All. The features that were enhanced in this study effectively improved the performance of the model over that achieved using the original set of features. The new features are Information Entropy (F5), and the enhanced features are Positive Content (F6), Negative Content (F25), and Known Malware (F25).

To conclude this section, Table 7 presents the five groups of features in this study. Several particular phenomena can be observed in the table. The model with positive behavior-based features does not identify benign scripts well, resulting in more FPs (false positives). The model with negative behavior-based features does not identify malicious scripts well, resulting in more FNs (false negatives). The All-Features model performs best with an accuracy of 99.27%, a precision of 99.52%, a recall of 99.04%, and an F1 score of 99.28% with the validation dataset.

Table 7. Performance summary for feature combination

Testing Features	Acc (%)	Precision (%)	Recall (%)	F1 Score (%)	AUC (%)
Characteristic-based	99.09	99.04	99.16	99.10	99.08
Positive behavior-based	92.07	86.73	**99.64**	92.74	91.95
Neutral behavior-based	96.46	98.99	94.00	96.43	96.50
Negative behavior-based	81.04	**100**	61.46	76.13	80.73
All features	**99.27**	99.52	99.04	**99.28**	**99.27**

5 Conclusions

This work proposed an identification and classification system, based on the XGBoost algorithm, for detecting malicious PowerShell scripts. Two types of features, characteristic-based and behavior-based features, are investigated. The behavior-based features can be subdivided into positive behavior-based, neutral behavior-based, and negative behavior-based. A total of 33 features are proposed for training models to identify malicious PowerShell scripts. Four features are enhanced; these are Information Entropy (F5), Positive Content (F6), Negative Content (F24), and Known Malware (F25). Feature analysis shows that the processing of features improves model performance, the accuracy of threat detection, and attack intent identification. In the future, the author will conduct research on unbalanced sets and repeated random cross-validations for improve the system effectiveness. Based on the experimental results in this study, three main goals are for future work are proposed; they are the collection of more sample data to increase the generalizability of the model; the proposal of an automated approach to feature extraction to reduce labor and time costs, and the application of the results herein to real-life situations to achieve the study's original purpose and to reduce the number of actual victims of, and the damage done by, malware.

References

[1] S. Wheeler, hananyajacobson, and shawnkoon, What is PowerShell?, Retrieved from https://docs.microsoft.com/en-us/powershell/scripting/ (last visited on 2022/06/30)

[2] D. Bohannon, Invoke-Obfuscation, Retrieved from https://github.com/danielbohannon/Invoke-Obfuscation/ (last visited on 2022/06/30)

[3] C. Wueest, D. Stephen, The increased use of powershell in attacks, Proc. CA, Symantec Corporation World Headquarters, 2016, pp. 1-18.

[4] D. Patten, The evolution to fileless malware, Retrieved from http://infosecwriters.com/Papers/DPatten_Fileless.pdf (last visited on 2022/06/30)

[5] VMware Carbon Black, 'PowerShell' Deep Dive: A United Threat Research Report, Retrieved from https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/docs/vmwcb-report-powershell-deep-dive.pdf (last visited on 2022/06/30)

[6] Trend Micro, Microsoft Detection Tools Sniff Out Fileless Malware, Retrieved from https://www.trendmicro.com/vinfo/us/security/news/cybercrime-and-digital-threats/microsoft-detection-tools-sniff-out-fileless-malware/ (last visited on 2022/06/30)

[7] Cisco Security Outcomes Study, Proven Success Factors for Endpoint Security, Retrieved from https://www.cisco.com/c/dam/en/us/products/collateral/security/2021-outcomes-study-for-endpoint.pdf (last visited on 2022/06/30)

[8] C. Beek, J. Fokker, D. McKee, S. Povolny, Trellix Advanced Threat Research Report: January 2022, McAfee Labs, 2022.

[9] Sudhakar, S. Kumar, An emerging threat Fileless malware: a survey and research challenges, *Cybersecurity*, Vol. 3, No. 1, pp. 1-12, January, 2020.

[10] A. Afreen, M. Aslam, S. Ahmed, Analysis of fileless malware and its evasive behavior, *Proceedings of the International Conference on Cyber Warfare and Security*, Islamabad, Pakistan, 2020, pp. 1-8.

[11] C. Liu, B. Xia, M. Yu, Y. Liu, PSDEM: a Feasible De-obfuscation Method for Malicious PowerShell Detection, *Proceedings of the IEEE Symposium on Computers and Communications*, Natal, Brazil, 2018, pp. 825-831.

[12] D. Ugarte, D. Maiorca, F. Cara, G. Giacinto, PowerDrive: accurate de-obfuscation and analysis of PowerShell malware, *Proceedings of the International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, Gothenburg, Sweden, 2019, pp. 240-259.

[13] D. Hendler, S. Kels, A. Rubin, Detecting Malicious Powershell Commands using Deep Neural Networks, *Proceedings of the Asia Conference on Computer and Communications Security*, Incheon, Republic of Korea, 2018, pp. 187-197.
[14] G. Rusak, A. Al-Dujaili, U. M. O'Reilly, Ast-based Deep Learning for Detecting Malicious Powershell, *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, Toronto, Canada, 2018, pp. 2276-2278.
[15] Z. Li, Q. Chen, C. Xiong, Y. Chen, T. Zhu, H. Yang, Effective and Light-weight Deobfuscation and Semantic-aware Attack Detection for Powershell Scripts, *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, London, United Kingdom, 2019, pp. 1831-1847.
[16] Y. Tajiri, M. Mimura, Detection of Malicious Powershell using Word-level Language Models, *Proceedings of the International Workshop on Security*, Fukui, Japan, 2020, pp. 39-56.
[17] Y. Fang, X. Zhou, C. Huang, Effective Method for Detecting Malicious PowerShell Scripts based on Hybrid Features, *Neurocomputing*, Vol. 448, pp. 30-39, August, 2021.
[18] J. Song, J. Kim, S. Choi, J. Kim, I. Kim, Evaluations of AI-based Malicious PowerShell Detection with Feature Optimizations, *ETRI Journal*, Vol. 43, No. 3, pp. 549-560, June, 2021.
[19] S. Choi, Malicious Powershell Detection using Graph Convolution Network, *Applied Sciences*, Vol. 11, No. 14, Article No. 6429, July, 2021.
[20] A. Alahmadi, N. Alkhraan, W. BinSaeedan, MPSAutodetect: A Malicious Powershell Script Detection Model Based on Stacked Denoising Auto-Encoder, *Computers & Security*, Vol. 116, Article No. 102658, May, 2022.
[21] Karttoon, PowerShellProfiler, Retrieved from https://github.com/pan-unit42/public_tools/tree/master/powershellprofiler/ (last visited on 2022/06/30)
[22] Y. H. Choi, T. G. Kim, S. J. Choi, C. W. Lee, Automatic Detection for Javascript Obfuscation Attacks in Web Pages through String Pattern Analysis, *Proceedings of the International Conference on Future Generation Information Technology*, Jeju Island, Korea, 2009, pp. 160-172.
[23] T. Chen, C. Guestrin, Xgboost: A Scalable Tree Boosting System, *Proceedings of the 22nd ACM Sigkdd International Conference on Knowledge Discovery and Data Mining*, San Francisco, California, USA, 2016, pp. 785-794.

Biographies

Hsiang-Hua Hung was born in Taiwan in 1998. She received the M.S. degree in electrical engineering from National Taiwan University of Science and Technology, Taipei, Taiwan, in 2022. Her main research interests include machine learning, feature engineering and malware detection.

Jiann-Liang Chen Prof. Chen was born in Taiwan on December 15, 1963. He received the Ph.D. degree in Electrical Engineering from National Taiwan University, Taipei, Taiwan in 1989. Since August 1997, he has been with the Department of Computer Science and Information Engineering of National Dong Hwa University, where he is a professor and Vice Dean of Science and Engineering College. Prof. Chen joins the Department of Electrical Engineering, National Taiwan University of Science and Technology, as a Distinguished professor and Dean now. His current research interests are directed at cellular mobility management, cybersecurity, personal communication systems and Internet of Things (IoT).

Yi-Wei Ma is an associate professor in National Taiwan University of Science and Technology. His research interests include Internet of Things, Cloud Computing, and Future Network.