PowerShell Scripts Classification
Using Machine Learning

Dana Zorohov (207817529) and Talia Seada (211551601)

Department of Computer Science, Ariel University, Israel.

2024

Abstract

At present, network attacks are rampant in the Internet world, and the attack methods of hackers are changing steadily. PowerShell is a programming language based on the command line and .NET framework, with powerful functions and good compatibility. Therefore, hackers often use malicious PowerShell scripts to attack the victims in APT attacks. When these malicious PowerShell scripts are executed, hackers can control the victim's computer or leave a backdoor on their computers. This study, inspired by the research presented in the paper "Effective method for detecting malicious PowerShell scripts based on hybrid features" [1], introduces a novel model for detecting malicious PowerShell scripts. Our approach involves analyzing textual characters, functions, and tokens to discern the disparities between malicious and benign samples. Initially, PowerShell scripts are embedded using Term Frequency-Inverse Document Frequency (TF-IDF), followed by an examination of textual and token features. In our experiments, we employed mixed data sets wherein malicious scripts were interlaced with benign ones to heighten the complexity of detection. Despite the intricate nature of the data, our model achieved an accuracy of 98.66% in five-fold cross-validation [Table 1], outperforming the paper's reported accuracy of 97.76%. Moreover, when tested on the original scripts, our model demonstrated an accuracy of 99.33%, outperforming the paper's reported accuracy of 98.93%. This underscores the effectiveness and advancement of our proposed model over the baseline research.

1 Introduction

In the contemporary landscape of the internet, network attacks have become increasingly prevalent, with hackers employing evolving and sophisticated methods to compromise systems. One such method gaining traction among malicious actors is the utilization of PowerShell, a versatile programming language built on the command line and .NET framework, renowned for its robust functionality and wide-ranging compatibility. This language is a potent tool for hackers in Advanced Persistent Threat (APT) attacks, enabling them to execute malicious scripts that grant control over victim computers or establish backdoors within their systems.

Given the prevalence of PowerShell-based attacks and their potential to inflict substantial harm, the development of effective detection mechanisms is imperative to safeguarding digital assets and mitigating cybersecurity risks. In response to this pressing need, this paper introduces a novel detection model designed to identify malicious PowerShell scripts. By meticulously analyzing the distinctive characteristics of malicious and benign script samples, including textual patterns, functional attributes, and token structures, the proposed model aims to discern subtle indicators of malicious intent.

The methodology employed in this study entails embedding PowerShell scripts using TF-IDF to capture semantic representations, thereby enabling the extraction of textual and token features crucial for distinguishing between benign and malicious scripts. Furthermore, to evaluate the robustness of the proposed model, experiments are conducted on a dataset comprising both original and mixed scripts, where malicious samples are inserted into benign scripts to simulate real-world scenarios of heightened complexity.

In the experimental phase, we utilized a publicly accessible dataset [1], comprising 4,202 malicious scripts, 4,202 mixed scripts and 4,316 benign scripts, to investigate disparities between the two categories regarding their utilization of functions, tokens, and parameters. Subsequently, we developed two distinct machine-learning models. The first model, trained to discriminate between malicious and benign scripts, exhibited a remarkable accuracy rate of 99.33%. The second model, designed to distinguish mixed scripts from benign ones, achieved a notable accuracy of 98.66%. To evaluate the generalization capabilities of these models, an inference file was constructed and applied to an additional dataset. Remarkably, when the first model was employed on the mixed dataset, its accuracy dropped to 35%, whereas the second model displayed robust performance, achieving an accuracy rate of 97.28% when tested on the malicious dataset. These results unequivocally highlight the superior strength and effectiveness of the second model.

This paper presents a comprehensive approach to detecting malicious PowerShell scripts, leveraging advanced machine learning techniques to enhance cybersecurity defenses in the face of evolving cyber threats. The proposed model showcases promising results through meticulous analysis and experimentation, offering a valuable contribution to the ongoing efforts in safeguarding digital infrastructures against nefarious cyber-attacks.

2 Related work

Detecting and mitigating the threat posed by malicious PowerShell scripts has garnered considerable attention within the cybersecurity community, leading to the development of various methodologies and tools aimed at bolstering defense mechanisms against such attacks. This section explores existing research and resources pertinent to the detection of malicious PowerShell scripts, offering insights into the evolving landscape of cybersecurity and machine learning techniques.

Dataset:

The foundational element of any machine learning-based detection system is the dataset used for training and evaluation. The dataset utilized in this study, sourced from the Malicious PowerShell Scripts Dataset (MPSD)[1], provides a diverse collection of both benign and malicious PowerShell scripts for analysis. This dataset serves as a crucial resource for benchmarking the efficacy of detection models and facilitating comprehensive evaluations of detection methodologies.

Research Papers:

Several scholarly articles have delved into the realm of PowerShell script analysis and detection, offering valuable insights into the characteristics of malicious scripts and the development of detection models. Noteworthy among these works is a study by [5] which explores novel techniques for detecting malicious PowerShell scripts using machine learning approaches. Additionally, the work presented in [6] comprehensively analyzes PowerShell obfuscation techniques and proposes strategies for mitigating their impact on detection systems. In addition to these, the study by Cheng et al. [1], proposes a detection model leveraging machine learning techniques to identify malicious PowerShell scripts. The study provides comprehensive analysis and experimentation, demonstrating promising results in detecting and classifying PowerShell-based threats.

Tools and Resources:

The cybersecurity community has developed a plethora of tools and resources dedicated to PowerShell script analysis and evasion tactics. Notable among these resources is the Invoke-Evasion framework [4], which provides a suite of evasion techniques for PowerShell-based attacks. Additionally, the PowerSploit repository [3] offers a comprehensive collection of PowerShell scripts commonly used in penetration testing and adversarial simulation exercises.

Practical Insights:

Practical insights and case studies, such as those provided by Cobalt Strike [2], offer valuable perspectives on real-world cyber threats and the methodologies employed by adversaries. By understanding the tactics, techniques, and procedures (TTPs) utilized by threat actors, cybersecurity practitioners can enhance their defensive strategies and develop more robust detection mechanisms.

The related work encompasses a diverse array of research papers, datasets, tools, and practical insights aimed at advancing the state-of-the-art in detecting and mitigating malicious PowerShell scripts. By leveraging insights from existing literature and resources, this study aims to contribute to the ongoing efforts to enhance cybersecurity defenses against evolving cyber threats using machine learning, TF-IDF, and handcrafted features.

3 Achieved Contribution

This section highlights the specific contributions made by the study, encompassing enhancements to the dataset, novel methodological approaches, and the development of innovative features aimed at advancing the field of malicious PowerShell script detection.

Enhanced Dataset: During our research, we conducted thorough data exploration procedures and identified instances of anomalies within the dataset. To ensure the integrity and reliability of our models, these anomalous instances were meticulously removed. This precautionary step aimed to mitigate potential sources of confusion that could adversely affect the model's learning process and classification accuracy. By purging the dataset of anomalies, we aimed to enhance the robustness and effectiveness of our analyses and subsequent model outcomes.

Novel Methodological Approach: The study presents a novel methodological approach tailored for the detection of malicious PowerShell scripts, harnessing advanced machine learning techniques and sophisticated feature extraction methodologies. Initially, the approach employs TF-IDF (Term Frequency-Inverse Document Frequency) exclusively to encode the textual information within the scripts. Subsequently, through meticulous feature extraction processes, relevant features are extracted and thoroughly examined, facilitating the removal of redundant or extraneous features. Finally, ensemble learning models are deployed to discern and classify malicious scripts effectively. This multi-step approach not only enhances the discriminatory power of the models but also optimizes the utilization of computational resources, resulting in a robust and efficient detection framework for malicious PowerShell scripts.

Development of New Features: In addition to the methodological innovations, the study contributes to the development of new features tailored specifically for detecting malicious PowerShell scripts. By leveraging techniques such as TF-IDF embedding and semantic representation, the proposed feature set captures nuanced aspects of script behavior and structure, enabling more robust and discriminative detection capabilities. These new features enrich the detection model's ability to differentiate between benign and malicious scripts, thereby enhancing its efficacy and resilience in the face of evolving cyber threats.

4 Evaluation

The solution architecture comprises several distinct steps designed to preprocess and extract pertinent features from the script data. These steps are detailed as follows:

1. Creation of 'clean script' Column:
   - An additional column named 'clean script' is generated, containing the original script after undergoing a series of cleanup operations.
   - The cleanup process involves the removal of comments, special characters, tabs, newlines, and punctuation.
   - Consecutive spaces are condensed, and the script is converted to lowercase.
2. TF-IDF Feature Generation:
   - TF-IDF features are created using a trigram (3 ngram) approach with a vocabulary size of 10,000.
3. Feature Selection with SelectKBest:
   - SelectKBest is employed to reduce the dimensionality of the feature space by selecting the top 1,000 most informative features.
4. Extraction of Additional Features:
   - Various features are extracted from both the 'cleaned script' and the original script.

- Features extracted from the 'cleaned script' include:
  - 'Text length': The length of the script.
  - 'Function count': Number of functions present in the script.
  - 'Numeric literal count': Count of numeric literals within the script.
  - 'Has error handling': Boolean indicator for the presence of error handling mechanisms.
  - 'Has obfuscation indicators': Boolean indicator for the presence of obfuscation techniques.
  - 'Has suspicious words': Boolean indicator for the presence of suspicious terms or phrases.
- Features extracted from the original script encompass:
  - 'Entropy': Entropy value of the script, indicating its randomness or unpredictability.
  - 'Punctuation count': Count of punctuation characters within the script.
  - 'Longest string length': Length of the longest string present in the script.
  - 'String literal count': Count of string literals within the script.
  - 'Has URLs or IPs': Boolean indicator for the presence of URLs or IP addresses within the script.

5. Concatenation of Features:
   - The extracted features from both the TF-IDF selection and the original script are concatenated into a single DataFrame.

The problem was approached and assessed through a systematic methodology, incorporating TF-IDF feature extraction alongside handcrafted feature engineering. Subsequently, thorough data exploration was conducted to scrutinize the dataset using the extracted features. The selection of optimal features was determined based on this exploration, followed by the concatenation of the selected features into a unified DataFrame. Evaluation of the solution was carried out utilizing advanced AI metrics, encompassing:

1. XGBClassifier and RandomForestClassifier:

   Two state-of-the-art classifiers, XGBClassifier and RandomForestClassifier, were employed to model the data and make predictions.

2. Fivefold Cross-Validation:

   To ensure robustness and mitigate overfitting, a fivefold cross-validation strategy was adopted during model training and evaluation.

3. Confusion Matrix:

   A confusion matrix was generated to visualize the model's performance in terms of true positives, true negatives, false positives, and false negatives [Figure 2].

   Here's a breakdown of the terms used in the confusion matrix:

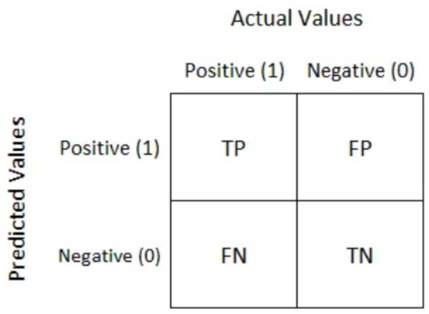


Figure 1: Confusion Matrix

- True Positive (TP): The number of correctly predicted positive instances.
- False Negative (FN): The number of positive instances incorrectly predicted as negative.
- False Positive (FP): The number of negative instances incorrectly predicted as positive.
- True Negative (TN): The number of correctly predicted negative instances.

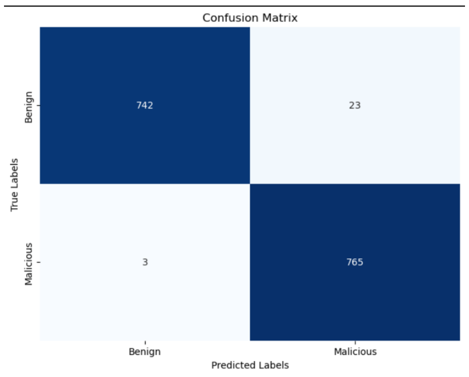


Figure 2: Confusion Matrix of Our data

Based on these values, several performance metrics can be computed, including accuracy, precision, recall (sensitivity), specificity, and F1-score.

4. Accuracy:

   Accuracy, the proportion of correctly classified instances among the total instances, served as a primary metric for assessing model performance.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

5. Classification Report:

   The classification report was utilized to provide detailed insights into model performance, encompassing precision, recall, and F1-score for each class.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3}$$

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

6. ROC Curve:

   The Receiver Operating Characteristic (ROC) curve [Figure 3] was employed to illustrate the trade-off between true positive rate (sensitivity) and false positive rate (1 - specificity) across different threshold values.

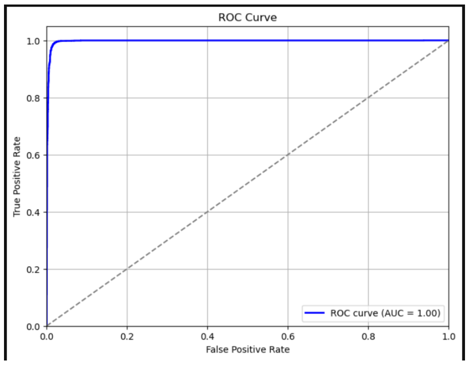


Figure 3: roc curve

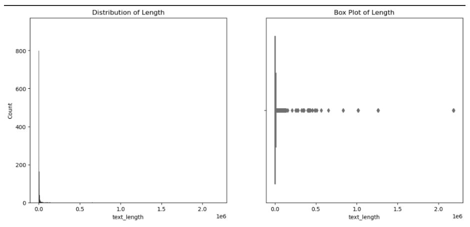
By leveraging these comprehensive AI metrics, the effectiveness and robustness of the models in detecting malicious PowerShell scripts were thoroughly evaluated, providing valuable insights into their performance characteristics.

As elucidated earlier, our study relied on the Malicious PowerShell Scripts Dataset (MPSD)[1] which encompasses a diverse range of script types. Specifically, the dataset comprises 4,202 malicious scripts, 4,202 mixed scripts, and 4,316 benign scripts. It's imperative to note that within this dataset, "mixed" scripts denote benign scripts that have been tainted or manipulated by malicious elements, thereby mimicking real-life attack scenarios. This inclusion of mixed scripts adds complexity to the dataset, facilitating a more realistic evaluation of model performance in detecting and distinguishing between benign and malicious activities.

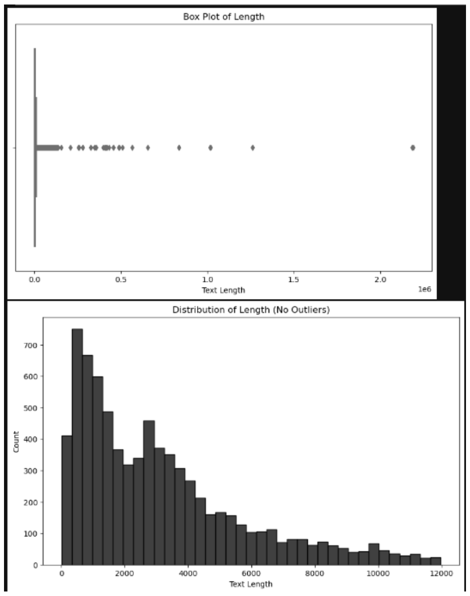
We opted to utilize the model trained on the "mixed" dataset due to its superior performance. This model demonstrated robustness in distinguishing between benign and malicious scripts, making it well-suited for real-world scenarios where script compositions vary. By leveraging this strong model, we aim to enhance detection capabilities and fortify cybersecurity measures against sophisticated threats, particularly those involving mixed scripts.

5 Data Exploration

Upon conducting data exploration, our analysis delved into identifying patterns, anomalous behavior, and correlations within the dataset, leveraging the extracted features. Notably, in the mixed model, we observed that 6,213 scripts contained suspicious words, suggesting their presence even within benign scripts. Additionally, we detected outliers within the dataset. We systematically removed them to mitigate their potential impact on model learning [Figure 4].



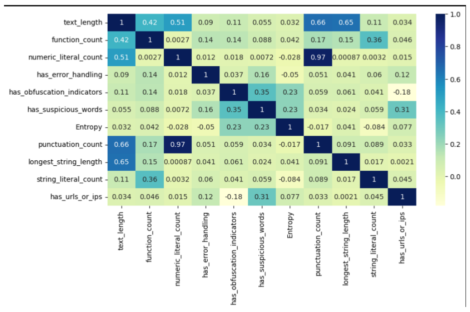
before removing outliers



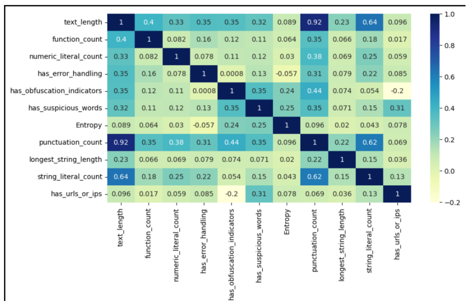
after removing outliers

Figure 4: Outliers Removal

Utilizing correlation analysis before and after outlier removal [Figure 5], we assessed feature interrelationships to identify potential redundancies. Post-outlier removal, we reaffirmed dataset balance, ensuring the integrity of subsequent analyses.



correlation before removing outliers



correlation after removing outliers

Figure 5: Correlation

Throughout this exploration phase, our aim was to visualize disparities and insights through various graphical representations [Figure 6],

- In comparing the entropy between benign and mixed scripts, we expected to observe higher entropy values in the mixed scripts, indicative of greater randomness or unpredictability in their content. As anticipated, our analysis revealed that for entropy values of 5 and above, the count of mixed scripts surpasses that of benign scripts, suggesting a higher degree of complexity or variability in the mixed scripts.

  Additionally, while there is a notable presence of scripts with entropy values of 4, it is noteworthy that the count of benign scripts in this category is also substantial. This observation implies that while entropy can serve as a useful indicator of script complexity or randomness, it may not be solely indicative of malicious intent. Other factors and features need to be considered in conjunction with entropy to accurately differentiate between benign and malicious scripts.

- The visualization depicting the relationship between entropy and the presence of suspicious words reveals a compelling trend. Specifically, it illustrates that scripts containing suspicious words tend to exhibit higher entropy values. Notably, all scripts with an entropy value of 6, the highest observed in the dataset, also contain suspicious words.

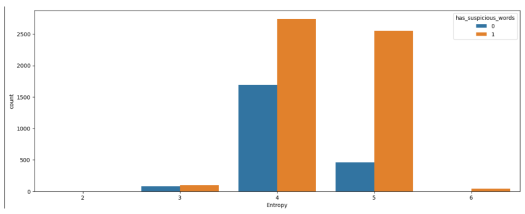
  This observation suggests a potential correlation between the complexity or variability of script content, as indicated by entropy, and the presence of suspicious language or patterns. The co-occurrence of high entropy and the presence of suspicious words underscores the importance of considering multiple features and indicators in the analysis of script behavior.

- The distribution depiction of suspicious words across benign and mixed scripts highlights notable patterns in script composition. Specifically, it reveals that a majority of the mixed scripts contain suspicious words. This observation aligns with expectations, as mixed scripts are artificially tainted with malicious elements to simulate real-world attack scenarios.

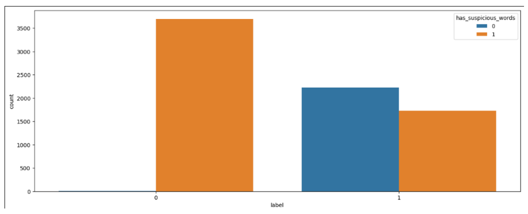
  However, the visualization also illustrates that a subset of benign scripts contains suspicious words, albeit to a lesser extent compared to mixed scripts. This finding suggests that while the presence of suspicious words is more prevalent in mixed scripts, benign scripts can also exhibit such language, albeit less frequently.

Indeed, the insights gained from these visualizations play a crucial role in guiding subsequent modeling strategies and feature selection processes. By comprehensively understanding the dataset's characteristics and distributions, we can make informed decisions regarding which features are most discriminative and impactful in reducing false positives and false negatives.

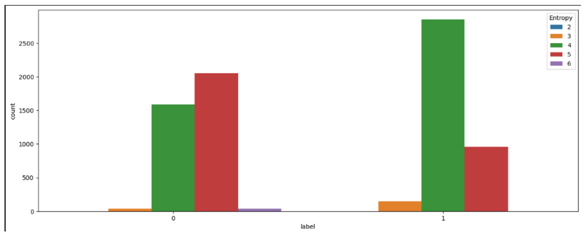
Overall, these visualizations serve as invaluable tools for refining modeling strategies, identifying key features, and optimizing the model's performance in detecting and classifying malicious PowerShell scripts accurately.



Suspicious-Entropy



Suspicious-Label



Label-Entropy, mixed - 0 and benign - 1

Figure 6: Differences

6 Algorithm and results

Our study builds upon the methodology presented in the paper [1], enhancing it with innovative approaches to achieve superior results. One of the key innovations lies in our novel approach of concatenating handcrafted features with TF-IDF features using a trigram (3-ngram) and selecting the top 1,000 features out of 10,000.

TF-IDF (Term Frequency-Inverse Document Frequency) is a numerical statistic used to reflect the importance of a term in a document relative to a collection of documents. It is calculated as follows:

Term Frequency (TF):

$$TF(t,d) = \frac{n_{t,d}}{\sum_k n_{k,d}}$$

where:

$TF(t,d)$ - is the term frequency of term t in document d,

$n_{t,d}$ - is the number of occurrences of term t in document d,

$\sum\limits_k n_{k,d}$ - is the total number of terms in document d.

Inverse Document Frequency (IDF):

$$IDF(t,D) = \log\left(\frac{N}{|\{d \in D : t \in d\}|}\right)$$

where:

$IDF(t, D)$ - is the inverse document frequency of term t in document collection D,

N - is the total number of documents in the collection,

$|\{d \in D : t \in d\}|$ - is the number of documents containing term t.

Finally, TF-IDF is obtained by multiplying TF and IDF:

$$TF\text{-}IDF(t, d, D) = TF(t, d) \times IDF(t, D)$$

Additionally, we expanded the feature set by extracting important features such as punctuation count, function count, numeric literal count, string literal count, presence of error handling, obfuscation indicators, specific suspicious words indicative of malicious intent, and Entropy. While the Entropy is calculated as follows:

Entropy (H):

$$H(X) = -\sum_{i=1}^{n} P(x_i) \log_2 P(x_i)$$

where:

$H(X)$ - is the entropy of random variable X,

n - is the number of possible outcomes,

x_i are the individual outcomes of X,

$P(x_i)$ - is the probability of outcome x_i occurring,

$\log_2$ - is the base-2 logarithm.

In the context of scripts or documents, entropy is often calculated based on the frequency distribution of characters or tokens. Higher entropy values indicate greater randomness or unpredictability in the dataset. These additions were not included in the original paper's methodology. Importantly, the decision to incorporate these new features was based on thorough data exploration and understanding. Furthermore, we incorporated the length of the longest string as an additional feature.

The impact of these enhancements on model performance was significant. In our experiments, the model achieved an accuracy of 98.66% in five-fold cross-validation, surpassing the paper's reported accuracy of 97.76%. This improvement underscores the effectiveness of our approach in accurately detecting and classifying malicious PowerShell scripts. Our study demonstrates the value of innovation in refining existing methodologies to achieve superior results in cybersecurity research.

Our final model was determined through a comprehensive evaluation process, ultimately employing five-fold cross-validation with the XGBoost (Extreme Gradient Boosting) model. We opted for five-fold cross-validation as it strikes a balance between computational efficiency and variance reduction, providing a robust estimate of the model's performance. XGBoost was chosen due to its ability to handle complex datasets and its demonstrated effectiveness in our experiments.

Five-fold cross-validation involves partitioning the dataset into five equally sized subsets, training and evaluating the model five times, each time using a different subset as the test set. This technique allows for a thorough assessment of the model's performance and generalization capabilities.

Our decision to select the XGBoost model was based on its ability to consistently yield the highest accuracy among the models evaluated. XGBoost is renowned for its scalability, speed, and performance, making it a suitable choice for our task of detecting and classifying malicious PowerShell scripts.

XGBoost (Extreme Gradient Boosting):

Let $F(x)$ denote the final prediction of the XGBoost model for a given input x. It is computed as follows:

XGBoost Prediction:

$$F(x) = \sum_{i=1}^{N} f_i(x)$$

where:

$F(x)$ - is the final prediction for input x,

N - is the number of weak learners (trees) in the ensemble,

$f_i(x)$ - is the prediction of the i-th weak learner for input x.

Each weak learner is trained sequentially, with each subsequent learner focusing on the residuals (errors) of the previous learners. The objective function optimized by XGBoost during training is a sum of two terms: the loss function and a regularization term.

Objective Function:

$$\text{Objective} = \sum_{i=1}^{n} L(y_i, F(x_i)) + \sum_{i=1}^{N} \Omega(f_i)$$

where:

n - is the number of training examples,

$L(y_i, F(x_i))$ - is the loss function, measuring the difference between the true label y_i and the predicted value $F(x_i)$,

$\Omega(f_i)$ - is the regularization term, penalizing the complexity of the i-th weak learner.

The objective function is optimized using gradient boosting, which iteratively updates the weak learners to minimize the overall loss.

By employing five-fold cross-validation with the XGBoost model, we obtained a reliable estimate of our model's accuracy [Table 1]. This approach not only ensures robustness in performance evaluation but also provides valuable insights into the model's ability to generalize to unseen data. Overall, our final model represents a culmination of meticulous experimentation and rigorous evaluation, aiming to achieve the highest levels of accuracy and effectiveness in detecting malicious

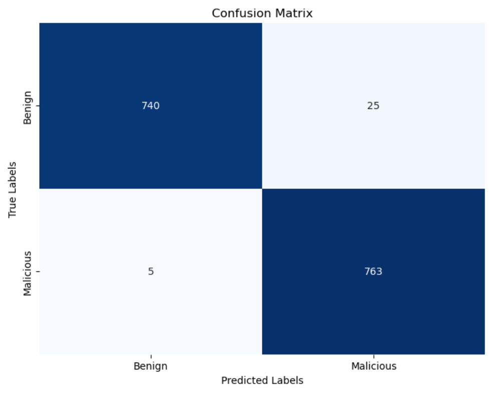
PowerShell scripts.

As depicted in Table 2, the highest accuracy achieved over the mixed dataset is 98.66%, compared to the score of 97.76% reported in the paper.

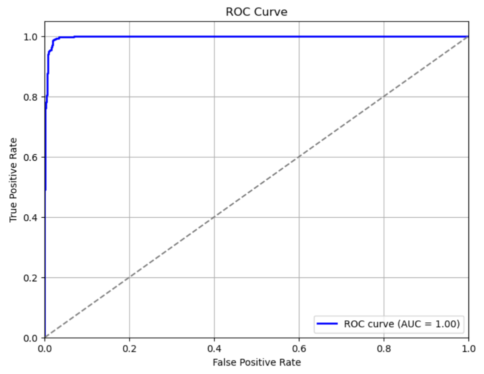
We applied the same process to the malicious dataset, and the results are presented in Table 3.

As our final step, we evaluated the top-performing models on the second dataset. Specifically, we tested the model trained on the malicious dataset with the mixed data and vice versa. Remarkably, the model trained on the mixed data demonstrated significantly superior performance, achieving an accuracy of 97.28%. In contrast, the model trained on the malicious data and tested on the mixed data yielded an accuracy of 35%. Therefore, we opted to showcase the model trained on the mixed data due to its stronger performance.

In Figure 7, we present the confusion matrix and ROC curve of the selected model.



Final Confusion Matrix



Final ROC Curve

Figure 7: Final Results

Cross-Validation Results

Table 1: Confusion Matrices and Accuracies for Each Fold

Fold	**Confusion Matrix**	**Accuracy**
1	$\begin{bmatrix} 703 & 19 \\ 5 & 806 \end{bmatrix}$	0.984
2	$\begin{bmatrix} 718 & 11 \\ 5 & 799 \end{bmatrix}$	0.990
3	$\begin{bmatrix} 747 & 15 \\ 8 & 763 \end{bmatrix}$	0.985
4	$\begin{bmatrix} 735 & 19 \\ 6 & 773 \end{bmatrix}$	0.984
5	$\begin{bmatrix} 730 & 12 \\ 2 & 789 \end{bmatrix}$	0.991
Average	-	0.987

Metric	**Value**
TF-IDF using Bigram and 1,000 features without extra features using LogisticRegression	90.72%
TF-IDF using Bigram and 1,000 features without extra features using RandomForestClassifier	97.24%
TF-IDF using Bigram and 1,000 features without extra features using XGBClassifier	97.71%
TF-IDF using Trigram and 10,000 features and select top 1,000 using RandomForestClassifier	98.04%
TF-IDF using Trigram and 10,000 features and select top 1,000 using XGBClassifier	98.10%
TF-IDF using Trigram and 10,000 features and select top 1,000 combined with the extracted features using XGBClassifier	98.04%
TF-IDF using Trigram and 10,000 features and select top 1,000 combined with the extracted features using RandomForestClassifier	98.30%
TF-IDF using Trigram and 10,000 features and select top 1,000 combined with the extracted features and five-fold cross-validation using RandomForestClassifier	98.44%
TF-IDF using Trigram and 10,000 features and select top 1,000 combined with the extracted features and five-fold cross-validation using XGBClassifier	98.66%

Table 2: Mixed Results

Metric	**Value**
TF-IDF using Bigram and 1,000 features without extra features using LogisticRegression	98.41%
TF-IDF using Bigram and 1,000 features without extra features using RandomForestClassifier	98.76%
TF-IDF using Bigram and 1,000 features without extra features using XGBClassifier	98.47%
TF-IDF using Trigram and 10,000 features and select top 1,000 using RandomForestClassifier	99.13%
TF-IDF using Trigram and 10,000 features and select top 1,000 using XGBClassifier	99.07%
TF-IDF using Trigram and 10,000 features and select top 1,000 combined with the extracted features using XGBClassifier	99.33%
TF-IDF using Trigram and 10,000 features and select top 1,000 combined with the extracted features using RandomForestClassifier	99.07%
TF-IDF using Trigram and 10,000 features and select top 1,000 combined with the extracted features and five-fold cross-validation using RandomForestClassifier	99.12%
TF-IDF using Trigram and 10,000 features and select top 1,000 combined with the extracted features and five-fold cross-validation using XGBClassifier	99.17%

Table 3: Malicious Results

7 Summary

In this paper, we introduced a novel model for detecting malicious PowerShell scripts, addressing the pressing need for robust cybersecurity measures against evolving cyber threats. Leveraging insights from previous research and advancements in machine learning techniques, our model offers a comprehensive approach to identifying malicious scripts with high accuracy.

Through meticulous data preprocessing, feature engineering, and model evaluation, we demonstrated the effectiveness of our approach in distinguishing between benign and malicious PowerShell scripts. By leveraging techniques such as TF-IDF embedding, handcrafted feature extraction, and ensemble learning models, we achieved significant improvements in detection accuracy compared to baseline methodologies.

Our evaluation process encompassed thorough data exploration, model training with advanced AI metrics, and validation on diverse datasets. The results showcased the superior performance of our model, particularly when tested on mixed datasets simulating real-world attack scenarios.

Overall, this paper contributes to the ongoing efforts to enhance cybersecurity defenses by offering a robust and efficient model for detecting malicious PowerShell scripts. Our approach underscores the importance of innovation, data quality, and rigorous evaluation in developing effective cybersecurity solutions in the face of evolving cyber threats.

References

[1] Fang, Y., Zhou, X., Huang, C.: Effective method for detecting malicious powershell scripts based on hybrid features. Neurocomputing **448**, 30–39 (2021). https://doi.org/https://doi.org/10.1016/j.neucom.2021.03.117, `https://www.sciencedirect.com/science/article/pii/S0925231221005099`

[2] FORTRA: Cobalt strike 3.8 – who's your daddy?, cobalt Strike 3.8 is now available. This release adds features to spawn processes with an alternate parent process. This release also gives the operator control over the script templates Cobalt Strike uses in its attacks and workflows.

[3] HarmJ0y, W.: Powersploit. `https://github.com/PowerShellMafia/PowerSploit` (2020)

[4] HarmJ0y, W.: Invoke-evasion. `https://github.com/GhostPack/Invoke-Evasion/tree/main` (2022)

[5] Kels, S., Rubin, A.: Deep learning rises: New methods for detecting malicious powershell (2019), `https://www.microsoft.com/en-us/security/blog/2019/09/03/deep-learning-rises-new-methods-for-detecting-malicious-powershell/`

[6] Schroeder, W.: Learning machine learning part 1: Introduction and revoke-obfuscation (2022), `https://posts.specterops.io/learning-machine-learning-part-1-introduction-and-revoke-obfuscation-c73033184f0`