SSH Shell Attacks

ANDREA BOTTICELLA*, Politecnico di Torino, Italy
ELIA INNOCENTI*, Politecnico di Torino, Italy
RENATO MIGNONE*, Politecnico di Torino, Italy
SIMONE ROMANO*, Politecnico di Torino, Italy

This paper introduces a comprehensive machine learning approach to analyze SSH shell attack sessions, leveraging both supervised and unsupervised learning techniques. Using a dataset of 230,000 unique Unix shell attack sessions, the methodology aims to classify attacker tactics based on the MITRE ATT&CK framework and uncover latent patterns through clustering. The key contributions of this work are:

- Development of a robust pre-processing pipeline to analyze temporal trends, extract numerical features, and evaluate intent distributions from large-scale SSH attack session data.
- Implementation of supervised classification models to accurately predict multiple attacker tactics, supported by hyperparameter tuning and feature engineering for enhanced performance.
- Application of unsupervised clustering techniques to uncover hidden patterns in attack behaviors, leveraging visualization tools and cluster analysis for fine-grained categorization.
- Exploration of advanced language models, such as BERT and Doc2Vec, for representation learning and fine-tuning to improve intent classification and session interpretation.

CCS Concepts: • **Computing methodologies** → **Supervised learning by classification**; **Unsupervised learning**; *Natural language processing*; *Machine learning*; *Machine learning approaches*; • **Security and privacy** → Intrusion detection systems.

Additional Key Words and Phrases: Machine learning, supervised learning, unsupervised learning, language models, text classification, clustering, intent classification, SSH shell attacks, security log analysis

Contents

1 INTRODUCTION

1.1 Motivation

Security logs play a crucial role in understanding and mitigating cyber attacks, particularly in the domain of network and system security. With the increasing sophistication of cyber threats, analyzing and interpreting security logs has become paramount for detecting, preventing, and responding to potential security breaches.

*The authors collaborated closely in developing this project.

Authors' Contact Information: Andrea Botticella, andrea.botticella@studenti.polito.it, Politecnico di Torino, Turin, Italy; Elia Innocenti, elia.innocenti@studenti.polito.it, Politecnico di Torino, Turin, Italy; Renato Mignone, renato.mignone@studenti.polito.it, Politecnico di Torino, Turin, Italy; Simone Romano, simone.romano2@studenti.polito.it, Politecnico di Torino, Turin, Italy.

Publication date: January 2025.

Unix shell attacks, especially those executed through SSH, represent a significant vector for potential system compromises.

The complexity of security log analysis stems from several key challenges:

- Logs are often unstructured and contain ambiguous or malformed text.
- Manual parsing and interpretation of logs is time-consuming and error-prone.
- The sheer volume of log data makes comprehensive manual review impractical.

These challenges underscore the need for automated, intelligent approaches to log analysis that can efficiently extract meaningful insights and identify potential security threats.

1.2 Objective

The primary objective of this research is to develop and evaluate machine learning techniques for automatic analysis and classification of SSH shell attack logs. Specifically, we aim to:

- Automate the process of log analysis and intent classification.
- Provide security professionals with insights into attack strategies.

The significance of this research lies in its potential to enhance cybersecurity threat detection and response capabilities by transforming complex, unstructured log data into actionable intelligence.

2 BACKGROUND

2.1 Security Logs and Attack Analysis

In the context of SSH shell attacks, logs document the sequence of commands executed during a malicious session, enabling security researchers to analyze attacker behaviors, techniques, and potential system impacts. However, the manual analysis of these logs is challenging due to their volume, complexity, and often non-standard formatting.

2.2 MITRE ATT&CK Framework

The MITRE ATT&CK (Adversarial Tactics, Techniques, and Common Knowledge) framework provides a comprehensive knowledge base of adversary tactics and techniques observed in real-world cyber attacks. This framework serves as a standardized methodology for understanding and categorizing attack strategies.
For our research, we focus on seven key intents derived from the MITRE ATT&CK framework:

- **Persistence:** Techniques used by adversaries to maintain system access across restarts or credential changes.
- **Discovery:** Methods for gathering information about the target system and network environment.
- **Defense Evasion:** Strategies to avoid detection by security mechanisms.
- **Execution:** Techniques for running malicious code on target systems.
- **Impact:** Actions aimed at manipulating, interrupting, or destroying systems and data.
- **Other:** Less common tactics including Reconnaissance, Resource Development, Initial Access, etc.
- **Harmless:** Non-malicious code or actions.

2.3 Research Approach

Our research employs a multi-faceted approach to SSH shell attack log analysis:

- Explore and preprocess a large dataset of Unix shell attack sessions.
- Apply supervised learning techniques to classify attack tactics based on session characteristics.
- Utilize unsupervised learning methods to discover patterns and clusters in attack sessions.
- Investigate the potential of advanced language models in understanding and categorizing attack intents.

3 DATA EXPLORATION AND PRE-PROCESSING

3.1 Introduction

This section outlines the steps taken to explore and preprocess the dataset used in this study. The primary objective is to understand the data characteristics, identify patterns, and prepare the data for further analysis. We focus on temporal trends, intent distributions, and textual features.

3.2 Dataset Overview

The dataset consists of logs from SSH attacks. Each entry includes the following features:

- **session_id**: An integer representing the unique identifier for each session.
- **full_session**: A string containing the full sequence of commands executed during the attack session.
- **first_timestamp**: The timestamp indicating the start of the session.
- **Set_Fingerprint**: A set of strings (labels) describing the nature of the attack.

Below is an example of the dataset structure:

	session_id	full_session	first_timestamp	Set_Fingerprint
0	0	[enable, system, shell, sh, cat, /proc/mounts,...	2019-06-04 09:45:11.151186+00:00	[Defense Evasion, Discovery]
1	1	[enable, system, shell, sh, cat, /proc/mounts,...	2019-06-04 09:45:50.396610+00:00	[Defense Evasion, Discovery]
2	2	[enable, system, shell, sh, cat, /proc/mounts,...	2019-06-04 09:54:41.863315+00:00	[Defense Evasion, Discovery]
...	...	...	...	...
233034	233046	[cat, /proc/cpuinfo, grep, name, wc, -l, echo,...	2020-02-29 23:59:22.199490+00:00	[Discovery, Persistence]

233035 rows × 4 columns

Fig. 1. Dataset

3.3 Dataset Preparation and Feature Extraction

The dataset used in this study was provided as a Parquet file.

To prepare the dataset for analysis and ensure its quality, several preprocessing steps were undertaken, described in detail below:

(1) **Decoding the full_session column**: The `full_session` column contained `90026` encoded shell scripts, making the raw data difficult to interpret. A decoding process was applied to these entries, converting them into a human-readable format.

(2) **Formatting the first_timestamp column**: The `first_timestamp` column was checked for consistency and converted into a standard datetime format.

(3) **Splitting the full_session column into lists of commands**: The `full_session` column contained entire attack sessions represented as single strings. Since we wanted each word / command to be a feature of our models, it was necessary to break it down into individual commands or keywords for more granular analysis. This was achieved through a multi-step splitting process using specific delimiters:

- The semicolon (`;`) was used to separate distinct commands within a session.
- The pipe (`|`) was used to divide concatenated commands or pipelines.
- Whitespace (` `) was used to further split commands into individual words or arguments.

This process allowed the identification of specific actions and parameters, facilitating a detailed analysis of the attack strategies.

(4) **Cleaning individual commands**: Each command or keyword extracted from the `full_session` column was processed to remove unnecessary or undesired elements. Regular expressions were used to strip unwanted symbols, and variable assignments.
(5) **Handling missing values**: To ensure the integrity and completeness of the dataset, a thorough check for missing values was performed across all columns.

```
enable ; system ; shell ; sh ; cat /proc/mounts ; /bin/busybox SAEMW ; cd /dev/
    shm ; cat .s || cp /bin/echo .s ; /bin/busybox SAEMW ; tftp ; wget ; /bin/
    busybox SAEMW ; dd bs=52 count=1 if=.s || cat .s || while read i ; do echo
    $i ; done < .s ; /bin/busybox SAEMW ; rm .s ; exit ;

[enable, system, shell, sh, cat, /proc/mounts, /bin/busybox, SAEMW, cd, /dev/shm
    , cat, .s, cp, /bin/echo, .s, /bin/busybox, SAEMW, tftp, wget , /bin/busybox
    , SAEMW, dd, bs, count, if, cat, .s, while, read, i, do, echo, i, done, .s,
    /bin/busybox, SAEMW, rm, .s, exit]
```

Listing 1. Dataset Processing

These preprocessing steps ensured that the dataset was well-structured, clean, standardized and ready for advanced analysis and machine learning tasks. By addressing issues such as encoding, formatting, and missing data, the preprocessing phase established a robust foundation for the study.

3.4 Temporal Analysis

The temporal analysis examines when the attacks were performed and the intent distribution over time.

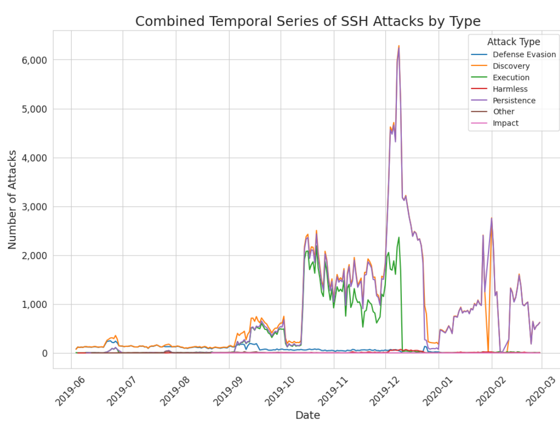

Fig. 2. Temporal Series of SSH Attacks

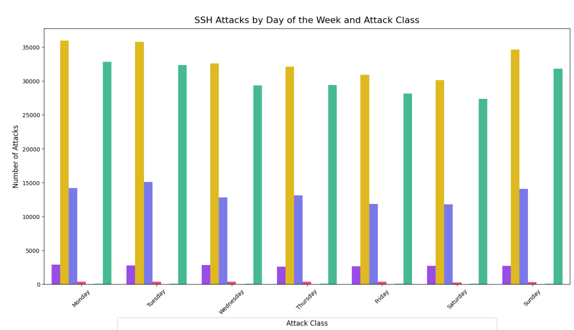

Fig. 3. Attacks by the day of the week

Our dataset spans from June 2019 to March 2020, revealing a notable concentration of attacks between mid-October 2019 and January 2020.

Further analysis was conducted on the distribution of sessions across various temporal dimensions (months, days of the week, etc.), but no significant patterns were identified.

3.5 Common Words Analysis

The analysis of common words within the attack sessions provides insights into frequently utilized commands and keywords. These elements are pivotal for understanding attacker behavior and informing model training for detecting attack intents.

To visualize and analyze the most frequent words, two approaches were used:

- **Word Cloud Visualization**: A word cloud (Figure 4) was generated to represent the most common words in the dataset. Commands like `grep`, `cat`, `echo`, and `rm` dominate the word cloud, highlighting their frequent usage in attack sessions.
- **Top 20 Most Common Words Bar Plot**: A bar plot (Figure 5) showcases the top 20 most frequent words, ordered by their frequency. This plot provides more quantitative detail, revealing the high occurrence of commands such as `grep` (over 1.2 million occurrences) and `cat` (over 1 million occurrences). Other significant commands include `echo`, `rm`, `uname`, and `name`. These commands are commonly associated with file manipulation, string searching, and process information, which are typical in malicious activities.

The combination of these visualizations supports the identification of commonly used commands and their potential roles in attack sessions.

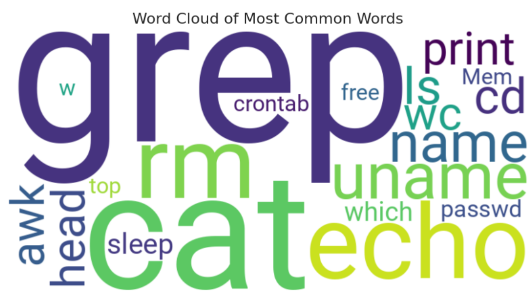

Fig. 4. Word Cloud of Most Common Words

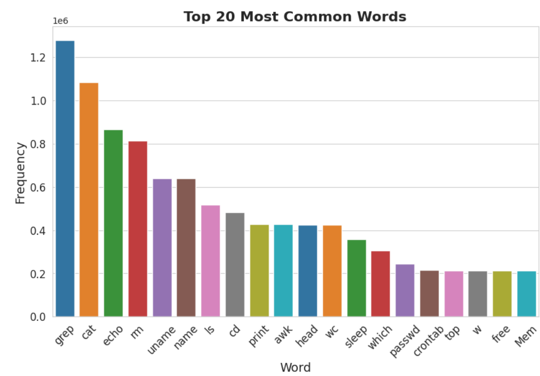

Fig. 5. Top 20 Most Common Words

3.6 Intent Analysis

This section presents an analysis of different intents identified in attack sessions and their relationships. Our analysis reveals distinct patterns in both the frequency of individual intents and their co-occurrence relationships, providing valuable insights into attacker behavior patterns.

Figure 6 illustrates the frequency distribution of different intents in our dataset. The analysis reveals that Discovery and Persistence are the most prevalent intents, with approximately 200,000 occurrences each. This suggests that attackers primarily focus on reconnaissance activities and establishing long-term presence in the targeted systems. Execution intent appears as the third most common, with roughly 100,000 occurrences, indicating a significant number of attempts to run malicious code. Defense Evasion shows notably fewer occurrences (around 20,000), while Harmless, Other, and Impact intents are relatively rare in the dataset.

The co-occurrence heatmap (Figure 7) reveals significant patterns in how different intents interact within attack sessions. Several key observations emerge:

- The strongest co-occurrence relationship exists between Discovery and Persistence (211,281 co-occurrences), suggesting that attackers often combine reconnaissance activities with attempts to maintain system access.

- Execution shows strong correlations with both Discovery (92,279) and Persistence (91,576), indicating that malicious code execution frequently accompanies both system discovery and persistence establishment attempts.
- Defense Evasion exhibits moderate co-occurrence with Discovery (18,825) and minimal correlation with other intents, suggesting that evasion techniques are primarily employed during reconnaissance phases.
- Harmless, Impact, and Other intents show minimal co-occurrence with other categories (with values mostly under 20 co-occurrences). This isolation pattern is partially explained by their low frequency in the dataset: Harmless with 2,206 occurrences, Impact with only 27 occurrences, and Other with 327 occurrences. These numbers represent a small fraction of the total recorded intents, naturally limiting their potential for co-occurrence with other intent types.

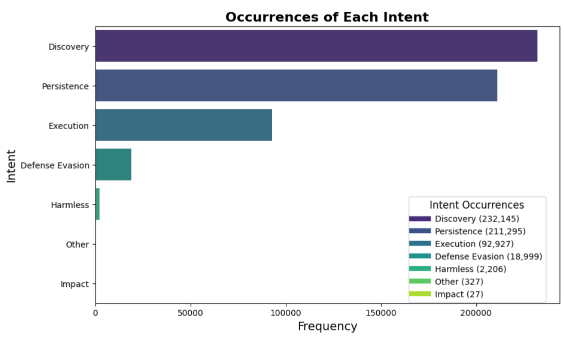

Fig. 6. Occurrences of each Intent

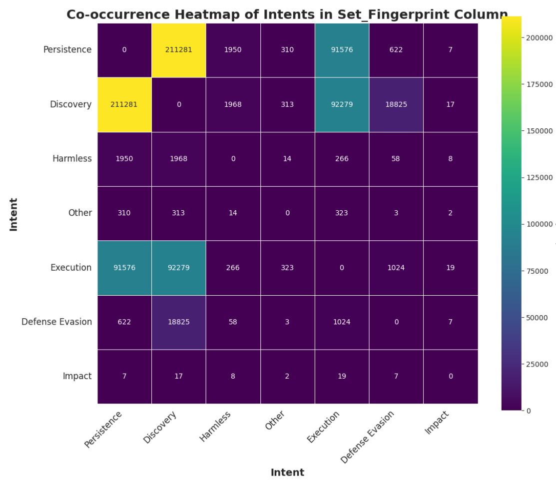

Fig. 7. Co-Occurrence Heatmap of Intents

These findings highlight common attack patterns where adversaries typically begin with discovery operations, followed by persistence establishment and execution of malicious code. This understanding can inform the development of detection strategies and defensive measures, particularly focusing on the most common intent combinations.

3.7 Session Analysis

The session analysis aims to understand the structural characteristics of the attack sessions in the dataset. This is accomplished through two visualizations: the Empirical Cumulative Distribution Function (ECDF) plots and the distribution of the number of words per session. The ECDF plots in Figure 8 provide insights into the distribution of the number of characters and words across all sessions.

- The ECDF for characters indicates that the majority of sessions have fewer than 10^3 characters, with a sharp increase between 10^2 and 10^3 characters. This suggests a high concentration of relatively short sessions.
- The ECDF for words shows a similar trend, with most sessions containing fewer than 10^2 words. The distribution reflects the concise nature of many attack sessions, potentially focusing on executing a limited number of commands.

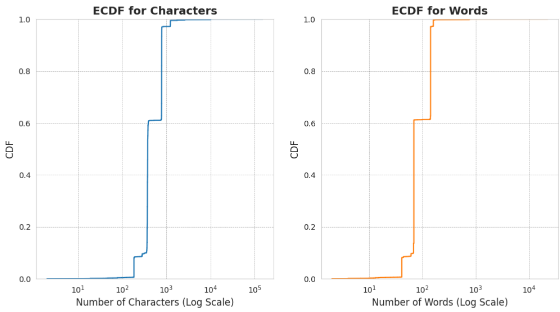

Fig. 8. ECDF for Characters and Words in Sessions

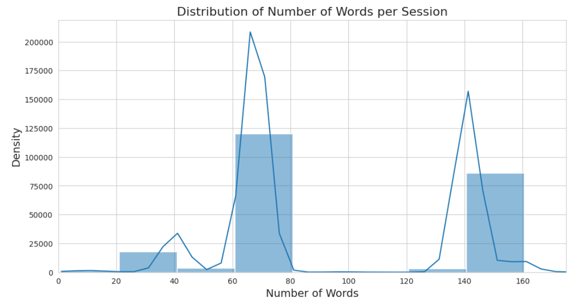

Fig. 9. Distribution of the Number of Words per Session

Figure 9 illustrates the density distribution of the number of words per session. Key observations include:

- The distribution is bimodal, with two distinct peaks. This likely reflects different categories of sessions, such as exploratory attacks with a larger number of commands versus simpler, targeted attacks with fewer commands.
- The majority of sessions have fewer than 100 words, reinforcing the compactness observed in the ECDF plots.
- Outliers with higher word counts likely represent complex sessions, potentially involving multiple stages or more sophisticated attack strategies.

The relevance of these plots lies in their ability to provide a foundational understanding of session structure, which is critical for feature engineering and model development. Shorter sessions may correspond to quick, automated attacks, while longer sessions might represent manual or multi-step intrusions. These insights can inform the design of models by emphasizing features tailored to the varying lengths and complexities of sessions, ultimately improving detection accuracy.

3.8 Text Representation

To enable further analysis of the dataset, the initial textual data was transformed into numerical representations. This step is essential for applying machine learning techniques, as numerical formats are required for model training and evaluation. Two widely used text representation techniques were employed:

- **Bag of Words (BoW)**: This technique represents each session as a vector where each dimension corresponds to a specific word, and the value represents the frequency of that word in the session. While simple and interpretable, BoW does not capture the importance or uniqueness of words across the dataset.
- **Term Frequency-Inverse Document Frequency (TF-IDF)**: This method extends BoW by weighting the word frequencies based on their inverse document frequency, thereby emphasizing words that are important within a session but less frequent across all sessions. This approach helps us, through the "min_df" parameter to remove from the initial dataset, the words that have a very low frequency. In our case, we set the "min_df" parameter to 0.01, which means that words that appear in less than 1% of the sessions are removed. In this way the features are reduced from 300.000 to 90.

Figure 10 and Figure 11 illustrate the transformed datasets using the BoW and TF-IDF techniques, respectively.

The TF-IDF representation was chosen for further analysis as it captures meaningful patterns by emphasizing word relevance, ensuring that subsequent analyses focus on the most distinctive features of each session, thereby enhancing the effectiveness of machine learning models in identifying patterns and predicting intents.

	Set_Fingerprint	enable	system	shell	sh	cat	proc	mounts	bin	busybox	...	20m	x17	go	x19	dota3
0	[Defense Evasion, Discovery]	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.000000	0.000000	0.000000	0.000559	0.000000
1	[Defense Evasion, Discovery]	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.000000	0.000000	0.000000	0.000559	0.000000
2	[Defense Evasion, Discovery]	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.000000	0.000000	0.000000	0.000559	0.000000
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
233034	[Discovery, Persistence]	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.001042	0.001611	0.002168	0.000000	0.001432

233035 rows × 90 columns

Fig. 10. Bag-of-Words

	Set_Fingerprint	enable	system	shell	sh	cat	proc	mounts	bin	busybox	...	20m	x17	go	x19	dota3
0	[Defense Evasion, Discovery]	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.000000	0.000000	0.000000	0.154131	0.000000
1	[Defense Evasion, Discovery]	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.000000	0.000000	0.000000	0.154131	0.000000
2	[Defense Evasion, Discovery]	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.000000	0.000000	0.000000	0.154131	0.000000
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
233034	[Discovery, Persistence]	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.067277	0.465673	0.133076	0.000000	0.066583

233035 rows × 90 columns

Fig. 11. TF-IDF

4 SUPERVISED LEARNING - CLASSIFICATION

4.1 Introduction

The supervised learning experiment in this project aimed to classify attack sessions into various intent categories derived from the ‘Set_Fingerprint‘ column of the dataset. This section explores the use of Random Forest and Support Vector Machines (SVM) as the primary models. The analysis focuses on how these models handle multi-label classification and evaluates their performance using metrics such as weighted F1-scores and confusion matrices.

This section outlines the model training and evaluation processes, and a detailed discussion of the results obtained. Each model's strengths and weaknesses are analyzed, providing insights into their application to multi-label classification problems.

4.2 Data Preprocessing

To effectively apply supervised learning models, it is crucial to represent the textual data in a numerical format. Raw text cannot be directly processed by most machine learning algorithms, so as we said in the previous section, we transformed the dataset into a structured numerical form.

Unlike BoW, which merely counts word occurrences without differentiating their relevance, TF-IDF down-weights frequently occurring words that may not carry significant information (e.g., common command-line syntax) and upweights rare but potentially more meaningful terms. This property helps improve the model's ability to differentiate between different types of attacks and intents.

To prepare the data for supervised learning:

(1) **Dataset loading**: To perform the supervised learning process, we decided to start from a TF-IDF representation of the dataset, thanks to which it was possible to assign a weight to each word of the SSH session attack.
(2) **Encoding Intents:** The ‘Set_Fingerprint‘ column was encoded into multi-label binary format using the MultiLabelBinarizer. Each intent was represented as a binary vector, allowing for simultaneous prediction of multiple labels.
(3) **Splitting the Data:** The dataset was divided into training (70%), validation (20%), and testing (10%) subsets. Stratified splitting was employed to ensure that all intents were well-represented in each subset.

These steps ensured the dataset was clean, balanced, and ready for supervised learning.

4.3 Model Training

Three models were trained and evaluated using their default configurations to establish baseline performance:

1. Random Forest
The Random Forest model was trained with default parameters, including 100 estimators and unlimited maximum depth. This initial training provided insights into potential overfitting or underfitting issues and served as a benchmark for subsequent tuning.

2. Support Vector Machines (SVM)
SVM was initially trained with default settings using a linear kernel and a regularization parameter $C = 1$. The performance was evaluated to assess the model’s ability to handle multi-label classification tasks with linearly separable data.

3. Logistic Regression
In order to have another view of the analysis, we performed the training with the logistic regression model. While the model performed well, the results were not as significant as those of RF and SVM, so they will not be discussed in detail in this section (see Appendix).

4.4 Evaluation Metrics

The models were evaluated using the following metrics:

- **Accuracy, Precision, Recall**
- **Confusion Matrices:** Provided insight into true positives, false positives, false negatives, and true negatives for each intent.
- **Weighted F1-Scores:** Measured the harmonic mean of precision and recall, with weights proportional to class support.

This evaluation allowed for the identification of baseline performance, highlighting potential areas for improvement through hyperparameter tuning.

4.5 Hyperparameter Tuning

To optimize each model, hyperparameter tuning was performed using a grid search approach. This process aimed to improve performance and address issues of overfitting or underfitting observed in the baseline models:

1. Random Forest
The grid search explored combinations of the number of estimators (50, 100, and 150) and maximum depth (unlimited, 50, and 100). The best-performing configuration was selected based on weighted F1-scores.

2. Support Vector Machines (SVM)
For SVM, the grid search varied the regularization parameter C (0.01, 0.1, 1, and 10) and the kernel type (linear and RBF). Additional tuning for the RBF kernel included the gamma parameter (scale and auto).

4.6 Results and Observations

4.6.1 Random Forest

- *Performance Overview with base model*

  The Random Forest base model showed robust performance, achieving high weighted F1-scores across all intents. However, minor overfitting was observed in intents with smaller sample sizes. Figure **??** illustrates the confusion matrix for the base model, showing excellent precision and recall for major intents like *Persistence* and *Defense Evasion*.

- *Hyperparameter Analysis*

  Hyperparameter tuning improved the Random Forest model's performance, particularly for minority intents. The optimized model reduced overfitting by limiting tree depth and increasing the number of estimators. Figure **??** highlights the improvement in weighted F1-scores across different configurations.

- *Comparative Analysis of Baseline and Optimized Models*

  Comparing the base and optimized models revealed significant improvements in precision and recall for minority intents. The optimized model maintained high performance for major intents while addressing issues of overfitting. Figure **??** presents the confusion matrix for the optimized Random Forest model.

4.6.2 Support Vector Machines (SVM)

- *Performance Overview with base model*

  The SVM base model performed well for linearly separable data, achieving high precision and recall for intents with large sample sizes. However, it struggled with minority intents due to its sensitivity to class imbalances. Figure **??** illustrates the confusion matrix for the base model.

- *Hyperparameter Analysis*

  Grid search tuning improved SVM's performance, especially for the RBF kernel. Increasing the regularization parameter C and fine-tuning the gamma parameter enhanced the model's ability to classify minority intents. Figure **??** shows the weighted F1-scores for different hyperparameter combinations.

- *Comparative Analysis of Baseline and Optimized Models*

  The optimized SVM model outperformed the base model, particularly for minority intents. Figure **??** displays the confusion matrix for the optimized model, highlighting the improvements in precision and recall for challenging classifications.

4.7 Conclusion

Random Forest and SVM both demonstrated strong performance in this experiment. Random Forest's ensemble nature provided robustness and generalization, making it the best-performing model overall. SVM, particularly with the linear kernel, offered comparable results but required more tuning to handle imbalanced classes effectively. These findings highlight the importance of selecting models and hyperparameters tailored to the dataset's characteristics and the problem's requirements.

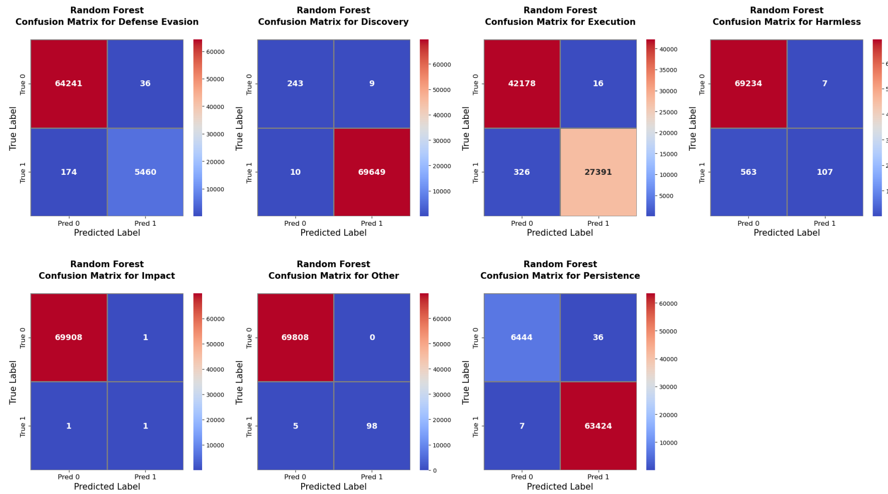

Fig. 12. Random Forest Confusion Matrices

==

PERFORMANCE ON TRAIN SET: Random Forest

==

Model	Set	Attack	Precision	Recall	F1-Score	Accuracy
Random Forest	Train	Defense Evasion	0.996432	0.986771	0.991544	0.997480
Random Forest	Train	Discovery	0.991428	0.995264	0.993338	0.999896
Random Forest	Train	Execution	0.996575	0.994876	0.995707	0.995887
Random Forest	Train	Harmless	0.963371	0.592712	0.652848	0.992208
Random Forest	Train	Impact	0.999991	0.940000	0.968081	0.999982
Random Forest	Train	Other	0.999991	0.993304	0.996625	0.999982
Random Forest	Train	Persistence	0.999491	0.997905	0.998697	0.999559

==

PERFORMANCE ON TEST SET: Random Forest

==

Model	Set	Attack	Precision	Recall	F1-Score	Accuracy
Random Forest	Test	Defense Evasion	0.995374	0.984278	0.989750	0.996996
Random Forest	Test	Discovery	0.980173	0.982071	0.981120	0.999728
Random Forest	Test	Execution	0.995873	0.993930	0.994879	0.995108
Random Forest	Test	Harmless	0.965265	0.579800	0.634430	0.991847
Random Forest	Test	Impact	0.749993	0.749993	0.749993	0.999971
Random Forest	Test	Other	0.999964	0.975728	0.987544	0.999928
Random Forest	Test	Persistence	0.999174	0.997167	0.998168	0.999385

Fig. 13. Random Forest Evaluation Metrics

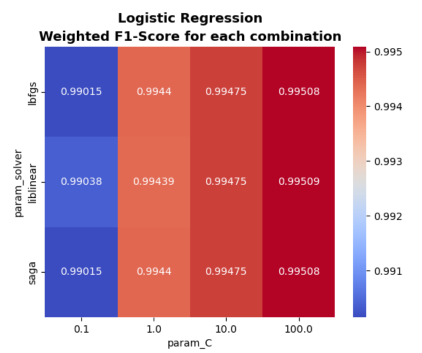

Fig. 14. Random Forest Weighted F1 Scores

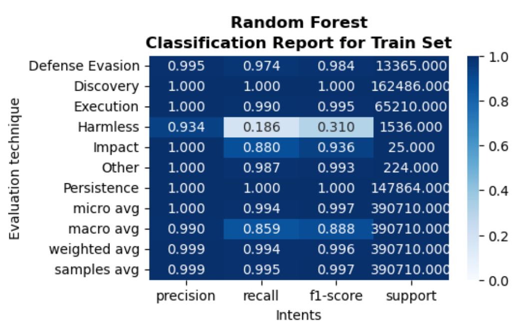

Fig. 15. Random Forest Classification Report Train Set

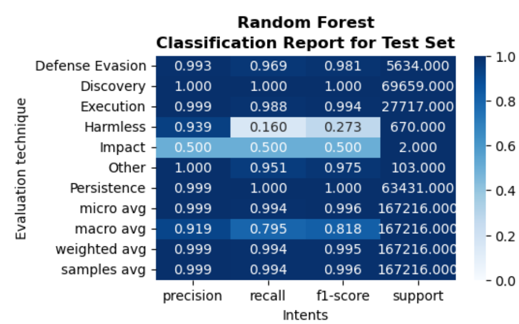

Fig. 16. Random Forest Classification Report Test Set

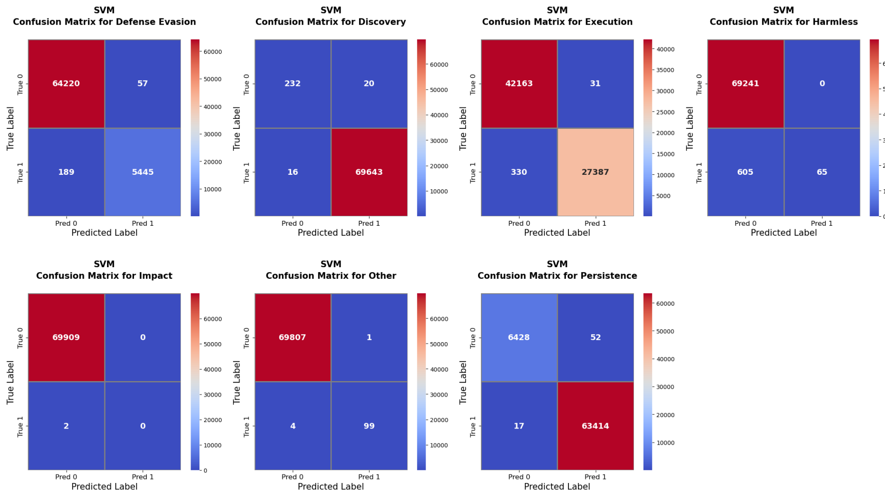

Fig. 17. SVM Confusion Matrices

PERFORMANCE ON TRAIN SET: SVM

Model	Set	Attack	Precision	Recall	F1-Score	Accuracy
SVM	Train	Defense Evasion	0.993578	0.984850	0.989167	0.996769
SVM	Train	Discovery	0.958840	0.964570	0.961687	0.999399
SVM	Train	Execution	0.995982	0.994316	0.995131	0.995335
SVM	Train	Harmless	0.992199	0.545895	0.581878	0.991442
SVM	Train	Impact	0.499923	0.500000	0.499962	0.999847
SVM	Train	Other	0.999966	0.975446	0.987397	0.999933
SVM	Train	Persistence	0.997561	0.997123	0.997342	0.999099

PERFORMANCE ON TEST SET: SVM

Model	Set	Attack	Precision	Recall	F1-Score	Accuracy
SVM	Test	Defense Evasion	0.993353	0.982783	0.987999	0.996481
SVM	Test	Discovery	0.967598	0.960203	0.963871	0.999485
SVM	Test	Execution	0.995552	0.993680	0.994595	0.994836
SVM	Test	Harmless	0.995669	0.548507	0.586260	0.991346
SVM	Test	Impact	0.499986	0.500000	0.499993	0.999971
SVM	Test	Other	0.994971	0.980575	0.987667	0.999928
SVM	Test	Persistence	0.998271	0.995854	0.997059	0.999013

Fig. 18. SVM Evaluation Metrics

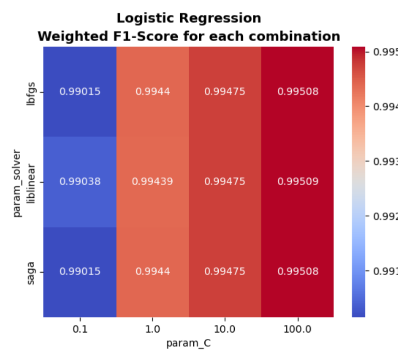

Fig. 19. SVM Weighted F1 Scores

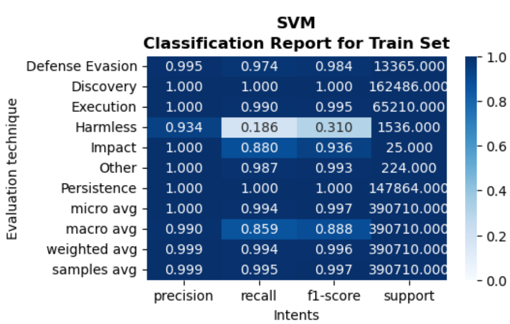

Fig. 20. SVM Classification Report Train Set

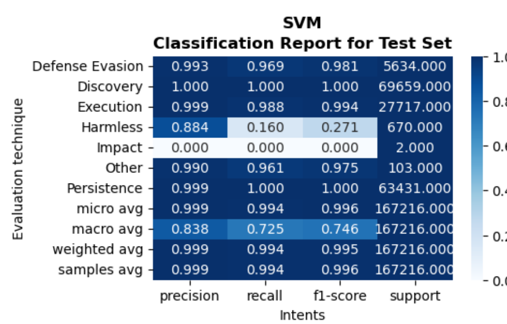

Fig. 21. SVM Classification Report Test Set

5 UNSUPERVISED LEARNING - CLUSTERING

5.1 Introduction

Unsupervised learning, a powerful branch of machine learning, was applied in this project to gain insights from SSH attack data. The primary focus was on leveraging clustering methods to group similar attack sessions based on their intrinsic patterns and characteristics. By analyzing these groups, the study aimed to uncover hidden relationships and categorize different attack intents and behaviors without relying on predefined labels.

5.2 Data Preparation

The dataset chosen was the one generated through the TF-IDF vectorization technique. This was made because it was essential to start with a dataset that represented in the best way the frequency and the importance of words, making each word as a dimension of our vector.

5.3 Clustering Methods

Clustering techniques were employed to uncover natural groupings within the dataset, providing insights into SSH attack patterns. The following methods were used:

- **K-Means Clustering**: The algorithm iteratively assigns each data point to the nearest cluster centroid and updates the centroids until convergence. The Elbow Method was applied to determine the optimal number of clusters by examining the total within-cluster sum of squares (inertia). Silhouette scores were also calculated to evaluate the cohesion and separation of clusters, ensuring the clustering results were meaningful and well-separated.
- **Gaussian Mixture Model (GMM)**: Unlike K-Means, GMM considers the probability of each data point belonging to a cluster, providing a more flexible and nuanced clustering approach. The optimal number of clusters was determined using a combination of log-likelihood scores, which measure how well the model fits the data, and silhouette analysis to validate cluster quality. This dual approach ensured the GMM provided reliable and interpretable clustering results.

5.4 Clustering Evaluation Techniques

5.4.1 K-Means Clustering

The evaluation of the K-Means clustering results is based on the Elbow Method and the Silhouette Score, which provide insights into the optimal number of clusters for the dataset.

The Elbow Method graph shows a steep decline in clustering error between 3 and 6 clusters, followed by a more gradual decrease as the number of clusters increases. The point of inflection, or "elbow," appears around 6 clusters, suggesting that adding more clusters beyond this point results in diminishing improvements in minimizing intra-cluster variance.

The Silhouette Score graph exhibits a rapid increase up to 5 clusters, reaching a stable high value of approximately 0.95. A drop is observed around 8 clusters, after which the score gradually increases again, peaking beyond 12 clusters. This pattern indicates that a smaller number of clusters (around 5-6) achieves a strong balance of cohesion and separation, while additional clusters beyond 12 continue to refine the structure with marginal improvements.

Considering both metrics, the optimal number of clusters for K-Means is likely between 5 and 6, ensuring a trade-off between clustering accuracy and computational efficiency.

5.4.2 Gaussian Mixture Model (GMM)

The evaluation of the GMM clustering results is based on the Silhouette Score and the Log-Likelihood Score, both of which provide insights into the quality of the clustering structure.

The Silhouette Score graph shows a sharp increase up to 5 clusters, reaching a stable high value around 0.95. A slight drop is observed at 8 clusters, followed by a steady increase, with the highest scores occurring beyond 12 clusters. This suggests that increasing the number of clusters generally improves separation and cohesion, though the optimal balance appears to be around 6 clusters, where the highest stable performance is first achieved.

The Log-Likelihood Score graph indicates a rapid increase from 3 to 5 clusters, after which the improvements become more gradual. Beyond 12 clusters, the score stabilizes, indicating diminishing returns in model fitting. This suggests that while increasing the number of clusters provides better representation of the data, the most significant improvements occur within the first few increments.

Considering both metrics, an optimal cluster configuration is likely between 6 and 8 clusters, balancing cluster separation, model likelihood, and computational efficiency..

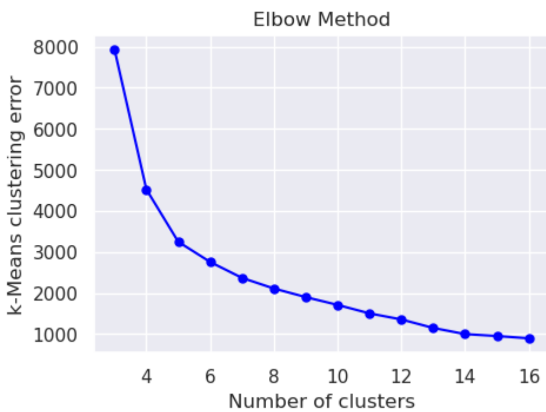

Fig. 22. K-means Elbow Method

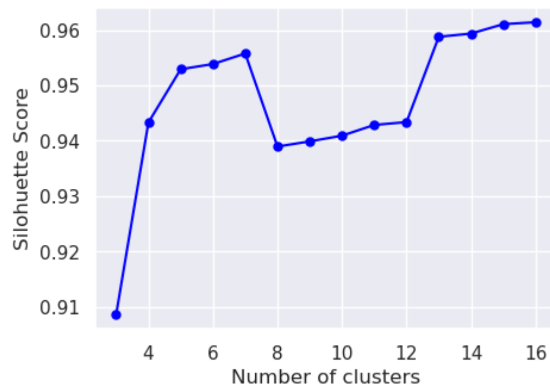

Fig. 23. K-means Silhouette Score

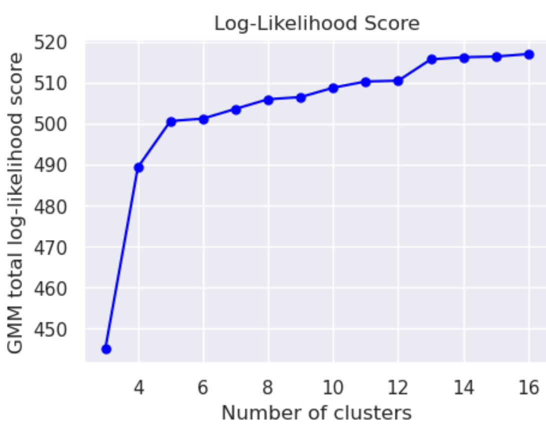

Fig. 24. GMM Log-Likelihood Score

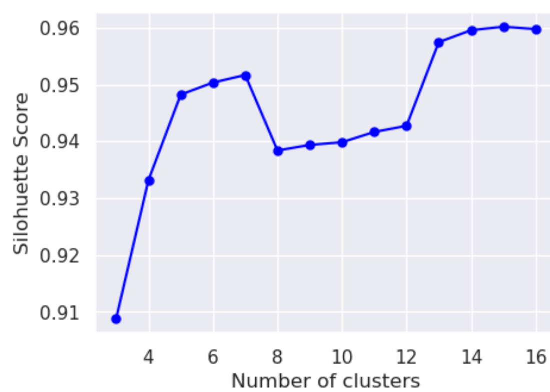

Fig. 25. GMM Silhouette Score

5.5 Hyperparameter Tuning

Hyperparameter tuning was conducted to optimize the performance of both clustering methods. For K-Means, parameters such as the initialization method (`k-means++` and `random`), the number of initializations, and the maximum number of iterations were fine-tuned using a grid search approach. Similarly, GMM parameters including the initialization method (`kmeans`), covariance type (`full` and `spherical`), and tolerance were optimized. These steps ensured the models were tailored to the dataset, resulting in better clustering outcomes.

5.6 Clusters Visualization

To visualize the clustering results, t-SNE dimensionality reduction was applied. Its functionalities makes it an excellent choice for visualizing clusters in datasets where direct interpretation is difficult due to high dimensionality. By projecting the data into a two-dimensional space, t-SNE enables us to identify patterns and groupings that may not be evident in the original feature space. The two-dimensional plots provided a clear representation of the clusters formed by both K-Means and GMM, highlighting their separability and internal consistency.

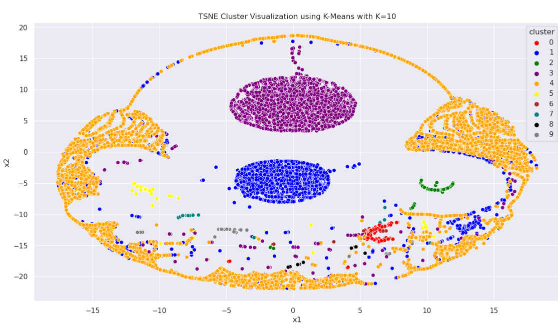

Fig. 26. t-SNE Visualization of K-Means Clusters

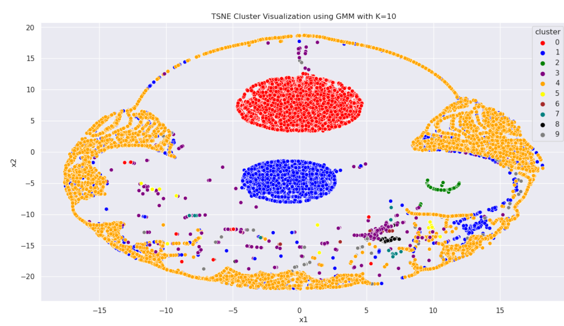

Fig. 27. t-SNE Visualization of GMM Clusters

5.6.1 *K-Means Visualization*

The t-SNE visualization provides a two-dimensional representation of the 10 clusters identified by the K-Means algorithm. This graph highlights the spatial distribution and separability of the clusters in the dataset.

Distinct cluster groupings are visible, with some clusters (e.g., the orange and purple clusters) forming highly compact and dense regions. This indicates strong intra-cluster similarity and effective separation from other clusters. Conversely, a few clusters (e.g., the red and yellow clusters) are more dispersed, suggesting potential overlap or variability in their features.

The visualization demonstrates that K-Means effectively partitions the dataset into meaningful groups, with well-defined boundaries for most clusters. However, the presence of dispersed clusters may reflect the complexity of certain patterns in the dataset, indicating that some attack behaviors share overlapping features. Overall, the t-SNE plot validates the clustering results by showing clear differentiation among the majority of the clusters.

5.6.2 *GMM Visualization*

The clusters exhibit distinct groupings, with some (e.g., the red and blue clusters) forming tightly packed regions that indicate strong intra-cluster similarity and minimal overlap. However, other clusters (e.g., the orange cluster) are more dispersed, reflecting the probabilistic nature of GMM, which allows for overlapping data points and accounts for uncertainties in cluster assignment.

Compared to the K-Means visualization, the GMM-based clusters appear more balanced in size and density. This suggests that GMM is effectively capturing the natural variability and overlapping characteristics within the dataset. Overall, the t-SNE plot validates the clustering results, showing that GMM provides a nuanced partitioning of the dataset while accommodating the complexities inherent in the data.

5.7 Clusters Analysis

5.7.1 Word Cloud Representation

Word clouds were generated for each cluster to highlight the most significant terms. These visualizations provided an intuitive understanding of the key features within each cluster by emphasizing frequently occurring terms. The approach helped in identifying the distinguishing characteristics of each cluster, offering insights into the behavioral patterns and intents associated with different attack sessions.

The word clouds revealed dominant keywords for specific clusters, such as commands, parameters, or phrases frequently used in SSH attacks. This information serves as a valuable reference for understanding the nature of the clustered attack sessions, aiding in further analysis and interpretation of the underlying data.

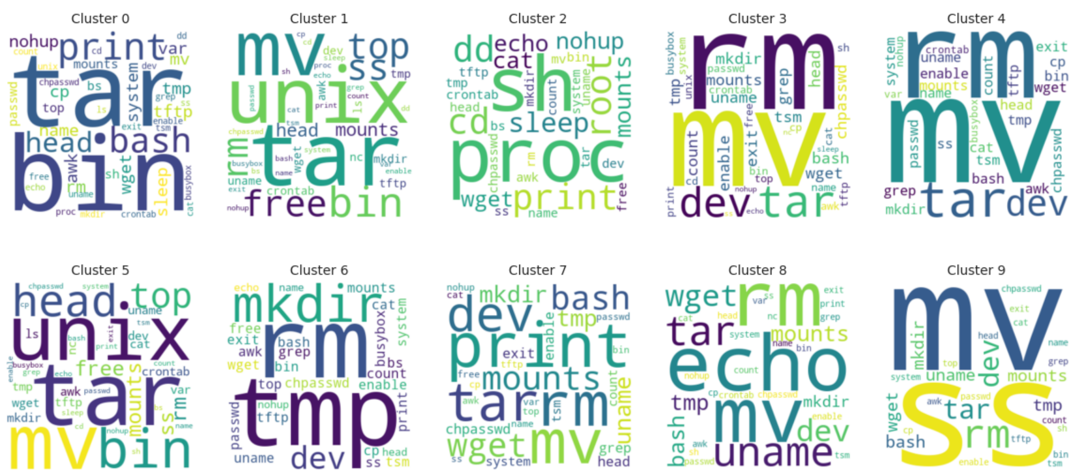

Fig. 28. Word Clouds for Each Cluster

5.7.2 Community Detection

Graph-based community detection was performed within selected clusters to identify subgroups. This approach involves constructing a graph where nodes represent features or sessions, and edges indicate relationships or co-occurrences within the data. The greedy_modularity_communities method was applied to detect communities, maximizing modularity to ensure well-defined groups.

The analysis revealed meaningful relationships and substructures within the clusters, helping to further refine the understanding of the dataset's internal patterns. These subgroups highlight finer-grained distinctions within the clusters, such as frequently occurring command sequences or behavioral traits common to specific attack types. Visualizations of these detected communities provide insights into the interconnectedness of the data and potential hierarchical structures in SSH attack patterns.

These visualizations demonstrate that distinct subgroups exist within Cluster 0. For K-Means, the graph reveals tight communities (e.g., green or pink) that suggest strong intra-community similarities, whereas more dispersed connections (e.g., cyan or orange) indicate weaker relationships. In contrast, the GMM graph shows balanced community sizes, reflecting GMM's probabilistic approach, which accounts for overlapping data points.

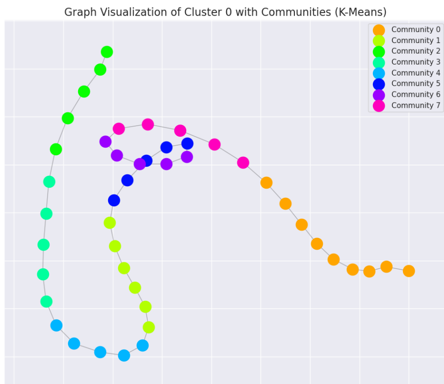

Fig. 29. Community Detection in Cluster 0 (K-Means).

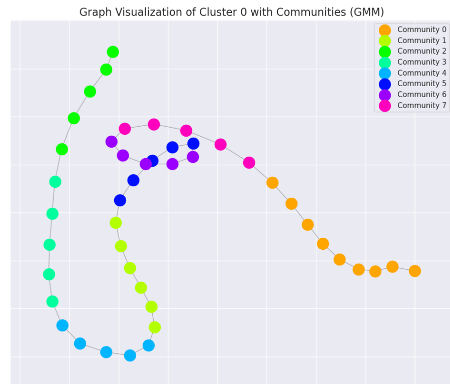

Fig. 30. Community Detection in Cluster 0 (GMM).

Overall, these detected communities provide finer-grained categorizations of attack patterns, enabling a deeper understanding of SSH attack behaviors beyond standard clustering methods.

5.8 Conclusion

This analysis provided valuable insights into the patterns of SSH attacks through clustering. The K-Means and GMM algorithms both effectively identified meaningful clusters, as supported by validation metrics and visualizations. Hyperparameter tuning further enhanced the performance of both models. The results demonstrate the potential of unsupervised learning in uncovering hidden patterns in complex datasets, providing a foundation for future applications such as anomaly detection and improved cybersecurity strategies.

6 LANGUAGE MODEL EXPLORATION

6.1 Introduction

This section documents the steps taken to fine-tune a pre-trained BERT model for multi-label classification using a dataset of SSH attack sessions. The focus was on customizing the final classification layer and training the model on the ‘Set_Fingerprint‘ intents. Performance metrics and visualizations are presented to evaluate the effectiveness of the model.

6.2 Data Preprocessing

The dataset was processed as follows:

- The ‘Set_Fingerprint‘ column was preprocessed for multi-label encoding using the ‘MultiLabelBinarizer‘ class, enabling the representation of intents as binary vectors.
- The dataset was split into 70% training, 20% validation, and 10% testing sets.

6.3 Tokenization

Pre-trained BERT’s tokenizer was used to convert text data into input IDs and attention masks. Tokenization results were saved to disk for reuse, reducing computational overhead.

6.4 Model Architecture

A pre-trained 'bert-base-uncased' model was used, with a custom linear classification head added to predict the intents. The classifier had an input size matching BERT's hidden layer dimensions and an output size equal to the number of intents.

6.5 Training Process

- The model was trained using the AdamW optimizer with a learning rate of 5×10^{-5} and a linear scheduler.
- A 'BCEWithLogitsLoss' loss function was applied for multi-label classification.
- Mixed precision training was used to enhance computational efficiency.
- Training was conducted for 4 epochs, using a gradient accumulation strategy to manage memory constraints.

6.6 Evaluation Metrics

- Metrics included precision, recall, F1-score, and ROC-AUC.
- ROC curves and training/validation loss curves were generated for each class.

6.7 Results

6.7.1 Training and Validation Loss

The training loss decreased steadily, indicating effective learning. However, the validation loss started increasing after epoch 2, suggesting potential overfitting. (See Figure 31).

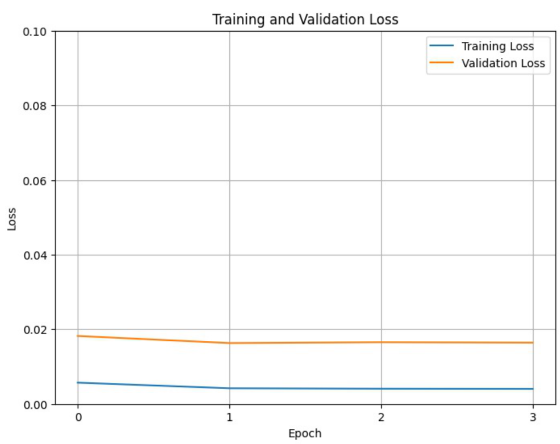

Fig. 31. Training and Validation Loss

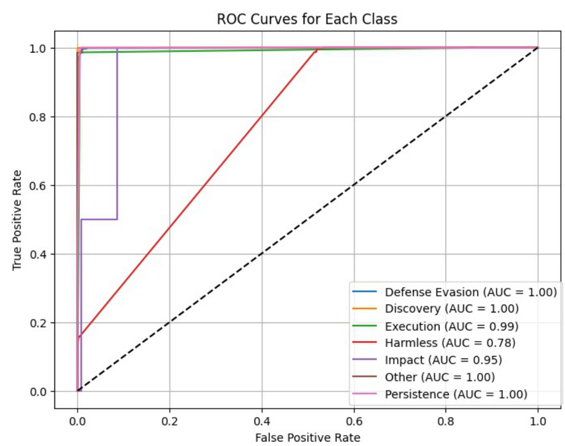

Fig. 32. ROC Curves for Each Class

6.7.2 ROC Curves

ROC curves were generated for each intent class. The area under the curve (AUC) values were as follows:

- **High AUC**: Discovery (1.00), Execution (0.98), Defense Evasion (0.97), Persistence (0.94).
- **Low AUC**: Harmless (0.77), Other (0.48), Impact (NaN due to lack of positive samples).

The variability in AUC values indicates that the model performs well on some intents but struggles with others. (See Figure 32).

6.7.3 *Test Metrics*
The model achieved the following metrics on the test set:

- Precision: 0.9974
- Recall: 0.9927
- F1-Score: 0.9937
- ROC-AUC: 0.9971

6.8 Discussion
The results demonstrate that the fine-tuned BERT model performs well for most classes, with high AUC scores for key intents like ‘Discovery‘ and ‘Execution‘. However, challenges remain:

- The model struggled with the ‘Other‘ class, likely due to limited or ambiguous data.
- Overfitting was observed after epoch 2, necessitating regularization techniques like dropout or early stopping in future iterations.

6.9 Conclusion
Fine-tuning a pre-trained BERT model proved effective for multi-label classification of SSH attack intents. While the model achieved reasonable overall performance, further tuning and data argumentation could address issues with minority classes and overfitting.

7 CONCLUSION

7.1 Summary of Key Findings
In this project, we explored various techniques for analyzing and classifying SSH shell attack logs. The primary objectives were to preprocess the data, perform exploratory data analysis, implement supervised and unsupervised learning models, and leverage advanced language models for classification tasks. Here, we summarize the key findings from each section of the project.

Data Exploration and Pre-processing: We began by loading and inspecting the dataset, identifying missing values, and handling duplicates. Temporal analysis revealed significant variations in attack frequencies over time, with notable peaks during specific hours and months. Feature extraction and common words analysis provided insights into the most frequent commands and intents used in the attack sessions.

Supervised Learning - Classification: We implemented and evaluated several machine learning models, including Logistic Regression, Random Forest, and Support Vector Machine (SVM). Hyperparameter tuning improved the performance of these models, and the result analysis highlighted the strengths and weaknesses of each approach. Feature experimentation with different text representation techniques, such as Bag of Words (BoW) and TF-IDF, demonstrated the impact of feature selection on model performance.

Unsupervised Learning - Clustering: Clustering techniques, such as K-Means and Gaussian Mixture Models (GMM), were used to group similar attack sessions. The elbow method and silhouette analysis helped determine the optimal number of clusters. Cluster visualization using t-SNE provided a clear representation of the clusters, and cluster analysis revealed common patterns and behaviors within each group.

Language Model Exploration: We explored the use of advanced language models, such as BERT, for classifying attack session tactics. Fine-tuning the pretrained BERT model on our dataset improved classification performance. Learning curves indicated the optimal number of epochs for training, helping to avoid overfitting.

7.2 Challenges Faced

Throughout the project, we encountered several challenges that required careful consideration and problem-solving.

Data Quality and Preprocessing: Handling missing values, duplicates, and inconsistencies in the dataset was a critical step. Ensuring the data was clean and well-prepared for analysis required significant effort. Additionally, the unstructured nature of the session text posed challenges for text representation and feature extraction.

Model Selection and Tuning: Selecting appropriate machine learning models and tuning their hyperparameters was a complex task. Balancing model complexity with performance and avoiding overfitting required iterative experimentation and validation.

Computational Resources: Training advanced language models, such as BERT, required substantial computational resources. Efficiently managing these resources and optimizing the training process was essential to achieve timely results.

Interpretability of Results: Interpreting the results of clustering and classification models, especially in the context of cybersecurity, was challenging. Ensuring that the findings were meaningful and actionable required careful analysis and domain knowledge.

7.3 Future Work

Based on the findings and challenges encountered in this project, we propose several directions for future work.

Enhanced Feature Engineering: Further exploration of feature engineering techniques, such as incorporating domain-specific knowledge and using advanced text representation methods, could improve model performance. Experimenting with additional features, such as network metadata and contextual information, may provide deeper insights into attack patterns.

Advanced Model Architectures: Exploring more advanced model architectures, such as transformer-based models and deep neural networks, could enhance classification accuracy. Transfer learning with other pretrained models and ensemble methods may also yield better results.

Real-time Analysis and Detection: Implementing real-time analysis and detection systems for SSH shell attacks could provide immediate insights and responses to potential threats. Integrating the models developed in this project into a real-time monitoring framework would be a valuable extension.

Broader Dataset and Generalization: Expanding the dataset to include a wider range of attack types and sources would improve the generalizability of the models. Collaborating with other organizations to share data and insights could enhance the robustness and applicability of the findings.

7.4 Conclusion

This project demonstrated the potential of machine learning and advanced language models for analyzing and classifying SSH shell attack logs. By leveraging various techniques, we gained valuable insights into attack patterns and behaviors, which can inform cybersecurity strategies and defenses. Despite the challenges faced, the results highlight the importance of data-driven approaches in enhancing cybersecurity threat detection and response capabilities. Future work in this area holds promise for further advancements and practical applications in the field of cybersecurity.