Politecnico di Torino

MACHINE LEARNING FOR NETWORKING

01DSMBG

Project Report
SSH Shell Attack session

Authors:

Hanane Kharbich - s319880
Tina Sayarmoafi - s307769
Masih Haghighatzadeh - s289712
Faezehalsadat Darbandi Kermanshah- s289880

Academic Year 2023/2024

Table of Contents

Introduction

In the realm of network security, machine learning (ML) offers promising solutions for detecting and analyzing threats, particularly in the context of Secure Shell (SSH) attacks. Our project utilizes a dataset comprising approximately 230,000 unique Unix shell attacks, captured via honeypot deployments, to explore and categorize attack tactics based on the MITRE ATT&CK framework. By applying various ML techniques, we aim to automate the identification and classification of malicious activities within these SSH sessions. This report details our methodology from data exploration to model application, including both supervised and unsupervised learning approaches, and explores the potential of language models to enhance threat detection. Through this analysis, we seek to demonstrate how ML can transform complex, text-based data into actionable security insights.

Section 1 – Data exploration and pre-processing

This section focuses on the initial exploration and pre-processing of the SSH shell attack dataset. We will analyze the temporal patterns of the attacks, investigate the nature of the commands within the sessions, and explore the distribution of words and intents across the dataset. Additionally, we will transform the textual data into a numerical format suitable for machine learning analysis. This preparatory work is essential for ensuring the effectiveness of our subsequent modeling efforts and for gaining insights into the characteristics and trends of the SSH attacks recorded in our dataset.

1.1 Temporal Series Analysis

Figure 1 illustrates the attacks distribution over time.

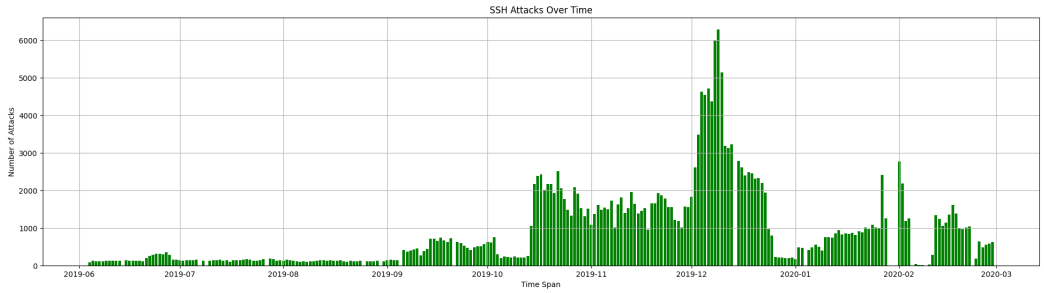


Figure 1: Distribution of attacks over time

The temporal analysis of SSH attacks within the dataset reveals significant insights into the distribution and intensity of attack occurrences over time. The bar plot illustrates clear fluctuations in the number of attacks, with pronounced peaks indicating periods of intense activity. Notably, the substantial peak around November 2019 suggests a period of heightened attack frequency, which could potentially indicate a coordinated attack campaign or the exploitation of newly discovered vulnerabilities.

In order to provide a smoother representation of the temporal distribution, the KDE is used as shown in figure 2

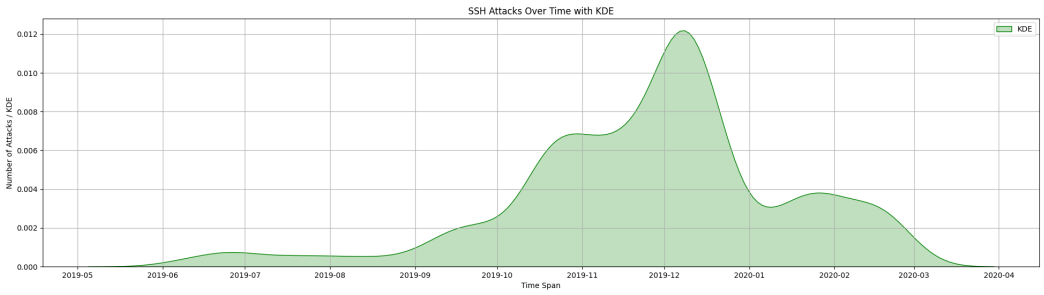


Figure 2: Probability density of the attacks over time

The Kernel Density Estimation (KDE) plot provides a smoothed visualization of the data, reinforcing the observations from the bar plot by highlighting the broader time periods during which attacks were more frequent. The sharp peak observed in the KDE plot around late 2019 emphasizes the surge in attack activity, offering a clearer understanding of when defenses might need to be especially fortified.

Moreover, figure 3 represents the CDF of the temporal distribution.

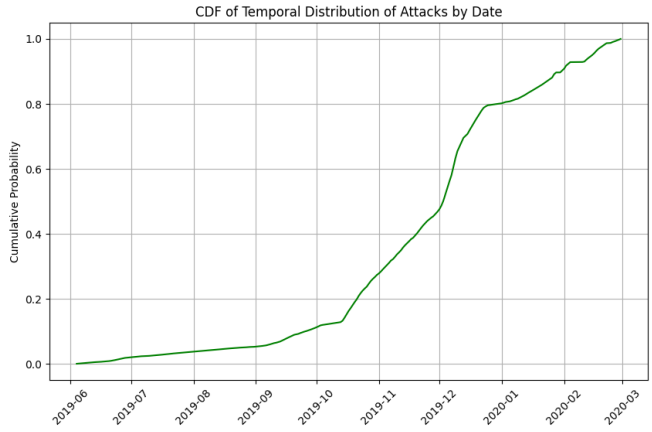


Figure 3: Cumulative distribution function of the temporal distribution over time

The Cumulative Distribution Function (CDF) offers additional perspective by depicting the cumulative probability of attacks over the timeline. The steep ascent observed in late 2019 corroborates the concentrated occurrence of attacks during this period, underlining that a significant proportion of the recorded attacks transpired in a relatively short time frame.

These findings demonstrate a non-uniform distribution of attacks across the observed timeline, with distinct periods of escalated activity.

1.2 Features of Attack Sessions

Figure 4 represents the empirical distribution of the number of characters in each session.

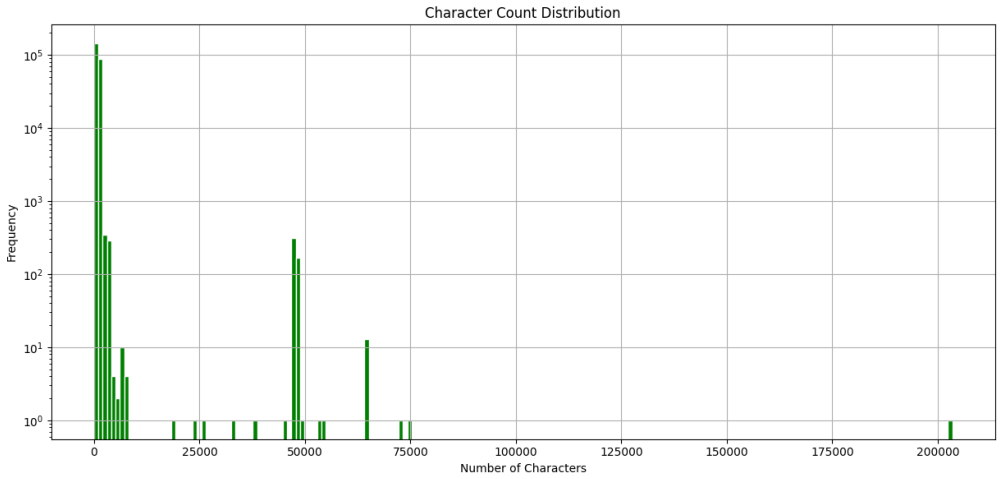


Figure 4: The empirical distribution of the number of characters in each session

The character count histogram displays a predominance of sessions with fewer characters, with the most frequent counts clustering at the lower end of the spectrum. There are, however, several peaks which suggest groupings of sessions with higher character counts. This could indicate different types of attacks or attackers who use more complex command sequences. The presence of sessions with extremely high character counts could be indicative of automated scripts or commands that involve substantial data manipulation or transfer.

Figure 5 represents the distribution of the number of word per session.

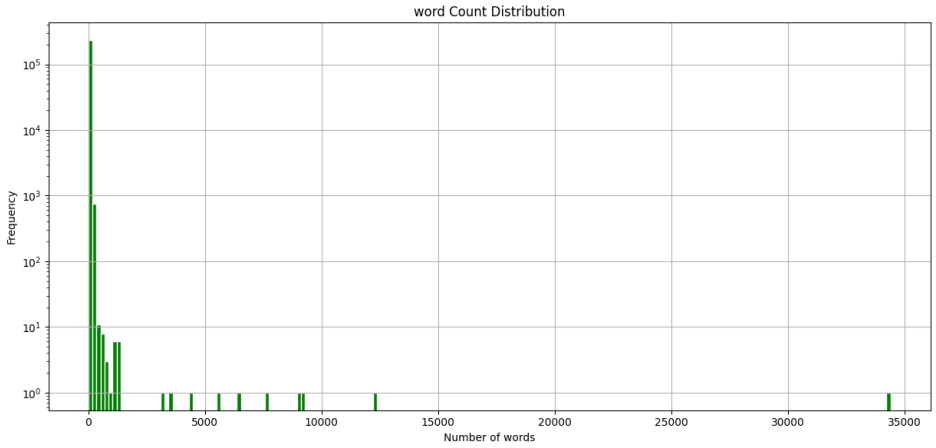


Figure 5: The distribution of the number of word per each session

Similar to the character count, the word count histogram also shows a right-skewed distribution with most sessions having fewer words. The distribution is sparse at higher word counts, highlighting that while most sessions are concise, a few contain a much more extensive command set. This variation in word count can be crucial for identifying more sophisticated attacks, which typically involve a higher number of commands and interactions with the system.

The observed distributions suggest that while many attacks are straightforward, involving fewer commands, there are distinctly more complex sessions that could potentially cause greater damage or indicate higher levels of attacker skill.

1.3 Most common words in the sessions

Figure 6 represent the frequency of the most 10 common word in the sessions.

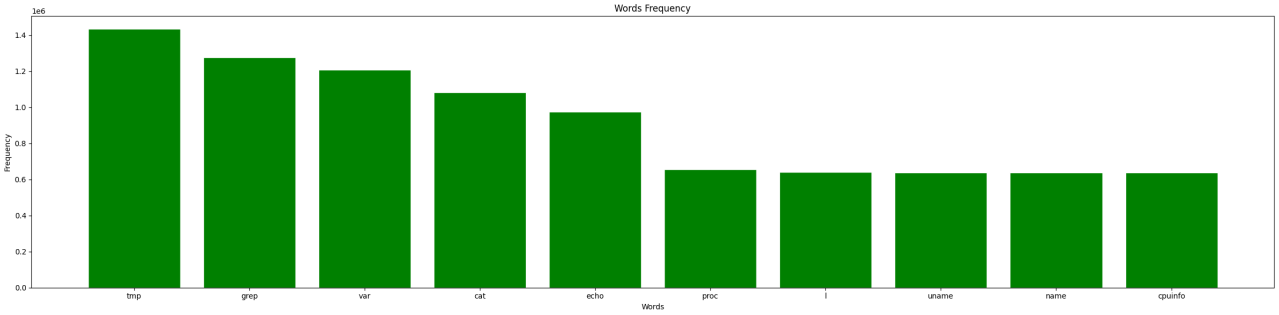


Figure 6: Most common words in the sessions

The analysis of common words in SSH attack sessions highlights frequent usage of specific Unix commands, such as tmp, grep, var, cat, echo, proc, uname, name, and cpuinfo. These commands are indicative of behaviors aimed at system exploration, information extraction, and exploitation. This pattern underscores the need for heightened monitoring of these commands to detect potential malicious activities early. Implementing strict usage policies and enhanced security measures around these frequently used commands can significantly strengthen defense mechanisms against SSH-based attacks.

1.4 Intents Distribution

Figure 7 represents the distribution of intents per session.

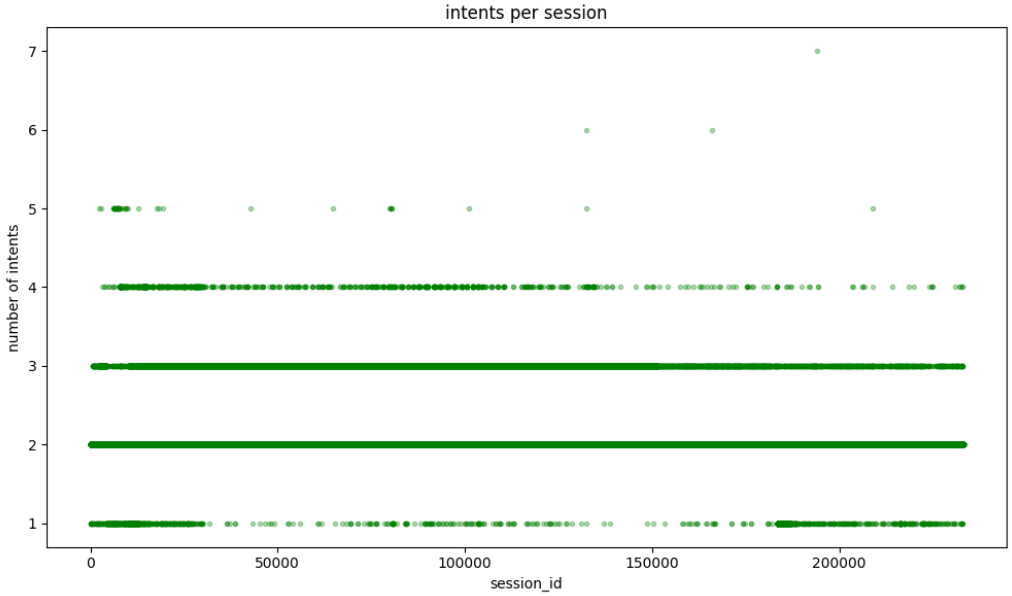


Figure 7: Intents per session

The scatter plot illustrates that the majority of SSH attack sessions consist of one to three intents, with a significant concentration around two intents per session. This pattern simplifies the identification of attack patterns. However, there are outliers with up to seven intents, indicating more complex attacks involving multiple tactics. This highlights the need for robust detection systems capable of handling these complex, multi-faceted attacks.

Figure 8 shows the frequency of intents.

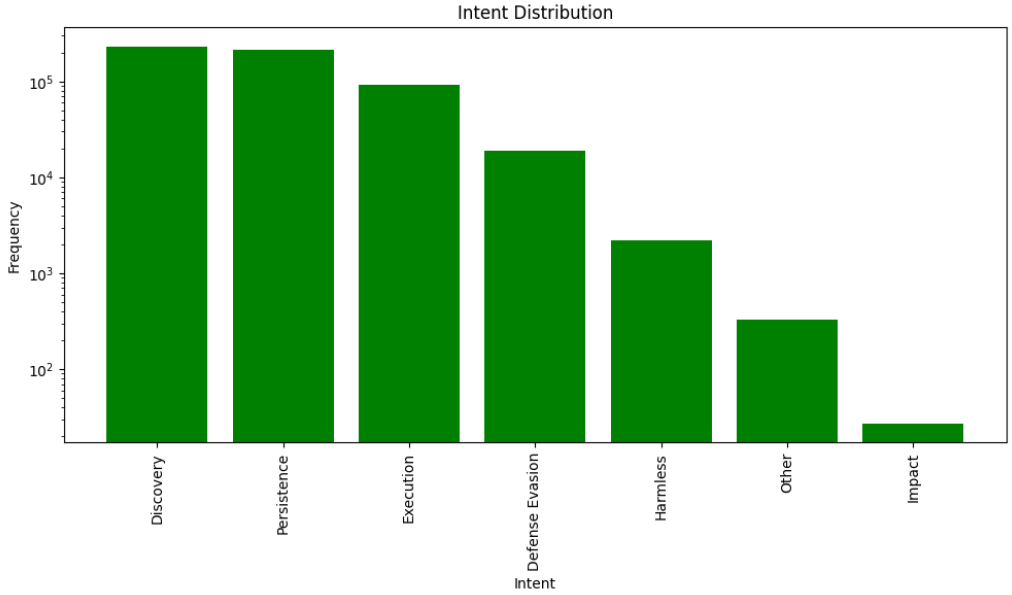


Figure 8: Intents frequency

The bar chart shows that Discovery, Persistence, and Execution are the most frequent intents, followed by Defense Evasion, Harmless, Other, and Impact. This predominance suggests that attackers are primarily focused on exploring the system, maintaining access, and executing malicious code. The lower frequency of Defense Evasion, Harmless, Other, and Impact indicates that while these intents are part of the attack strategy, they are not as prevalent.

Figure 9 illustrates the distribution of intents over time.

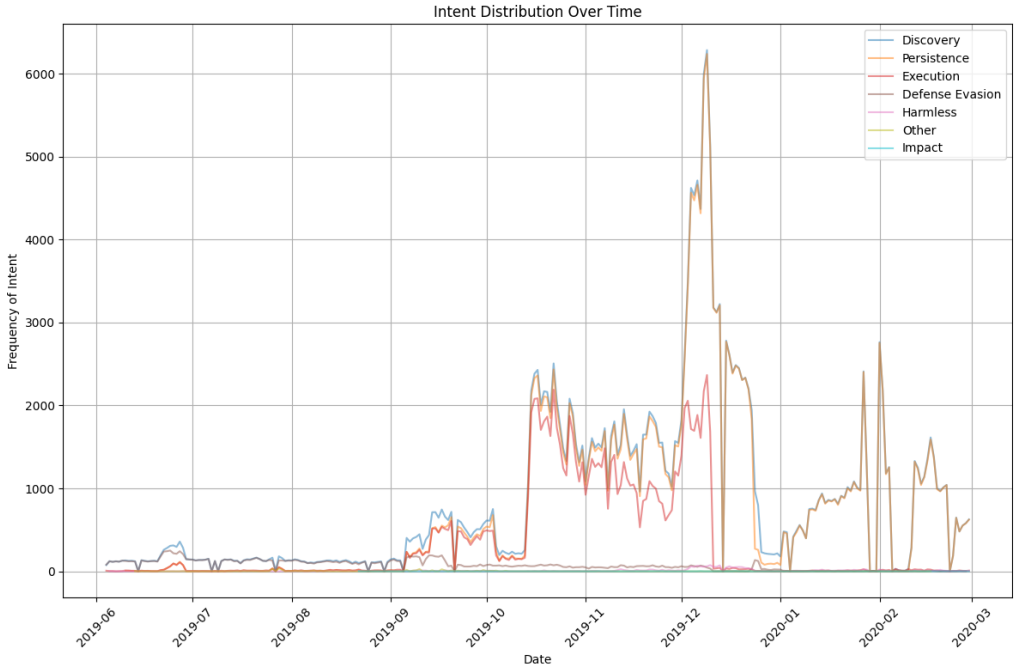


Figure 9: Intents distribution over time

The line plot indicates fluctuations in the frequency of various intents over time, with significant peaks around late 2019. This temporal analysis reveals that the frequency of attack intents can vary considerably over time, with spikes possibly corresponding to specific attack campaigns or the exploitation of newly discovered vulnerabilities. This emphasizes the need for continuous monitoring and adaptive security measures to respond to evolving threats.

1.5 Bag of words (BoW)

The SSH Shell Attack session data is converted into a numerical format using the Bag of Words (BoW) model through Scikit-Learn's CountVectorizer. This method effectively tokenized session commands into a sparse matrix representation, essential for the subsequent application of machine learning techniques.

1.6 Term Frequency-Inverse Document Frequency

TF-IDF (Term Frequency-Inverse Document Frequency) is employed to numerically represent SSH shell attack sessions, enhancing the recognition of important terms beyond mere frequency counts. We implemented custom pre-processing to handle command parsing issues and excluded common Linux commands through a tailored stop words list. The TF-IDF values, represented in a sparse matrix, were normalized to standardize data scaling, crucial for subsequent machine learning analysis. This process highlights significant terms within the sessions, supporting more effective detection and classification of attack patterns.

Section 2 – Supervised learning – classification

This section delves into the application of supervised learning techniques to classify SSH shell attacks based on the tactics observed. We aim to decipher the intent behind each attack session by training models to associate textual data with specific attack categories. Given the multi-label nature of our dataset, where each session could represent multiple tactics, we approach this problem by decomposing it into several binary classification tasks. We will evaluate different machine learning algorithms, analyze their performance, and explore the potential of model tuning to enhance accuracy. This section not only tests the feasibility of classifying complex attack patterns but also seeks to identify the most effective methods for practical deployment in cyber-security environments.

2.1 Dataset preparation for supervised learning

The dataset is prepared for supervised learning by splitting it into training and test datasets using an iterative train-test split, ensuring a proportional representation of labels. We combined 'first_timestamp' with expanded 'tf_idf' features into a single feature set, which was then standardized using a StandardScaler. This standardization, important for ensuring consistent input feature scales across our models, included both the numerical 'firs_timestamp' and the TF-IDF vectors, setting a solid foundation for accurate model training and evaluation.

2.2 Model training

In this phase two models have been chosen, the **Decision Tree Classifier** and the **Random Forest Classifier**.

- **Decision Tree Classifier:**
  - **Performance Evaluation:**
    The performance of a Decision Tree classifier is assessed within a Binary Relevance framework on our SSH shell attack dataset. The model demonstrated high training accuracy of approximately 99.95 % and a test accuracy of 97.07 %, suggesting some over-fitting, as it may be too finely tuned to the training data. The multi-label confusion matrix highlighted varying levels of model sensitivity and specificity across different labels, with some labels showing excellent specificity but lower sensitivity. This variation in performance across labels and the observed over-fitting indicate a need for further model tuning to enhance generalization on unseen data.
    Moreover, the Decision Tree classifier demonstrated excellent performance metrics: a Hamming Loss of 0.0049 indicates a low rate of incorrect label assignments; a Precision of 0.9919 confirms the accuracy of positive predictions; a Recall of 0.9946 shows the model's ability to identify relevant instances effectively; and an F1-score of 0.9932 signifies a strong balance between precision and recall. These results highlight the model's robustness and reliability in classifying SSH shell attack intents, making it highly effective for practical security applications.

  - **Confusion Matrix:**
    Moreover, Figure 10 shows that the Decision Tree classifier demonstrated robust performance in recognizing common attack intents such as Execution and Persistence, with high accuracy and minimal mis-classifications. However, it faced challenges in accurately classifying rarer intents like Impact, showing a need for improved sensitivity. The model also exhibited a tendency to mis-classify non-malicious actions as attacks in the Harmless and Defense Evasion categories, indicating an area for potential refinement. This analysis underscores the classifier's strengths in identifying frequent attack patterns and highlights opportunities for enhancing its precision and reliability across varied attack scenarios.

  - **Classification Report:**

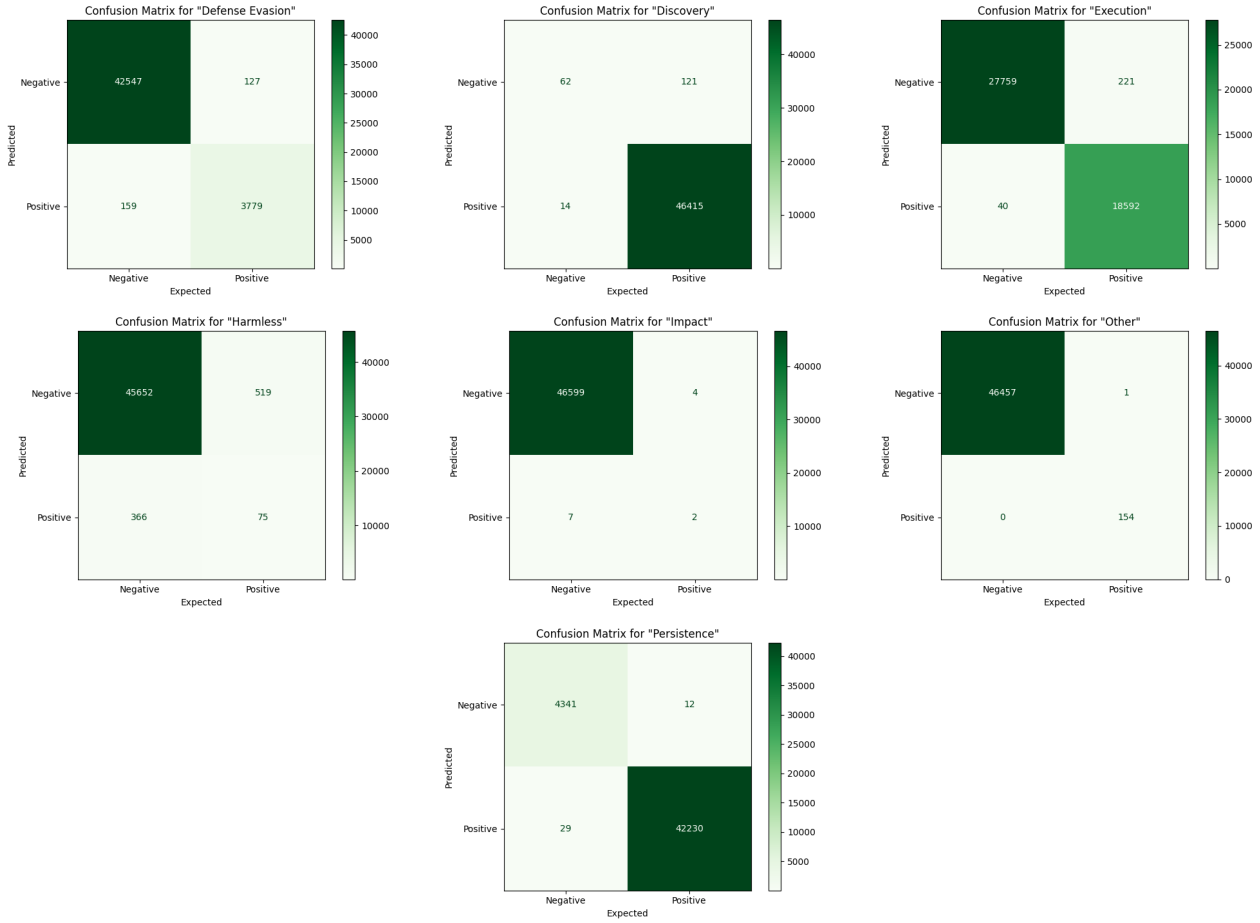


Figure 10: Confusion Matrices - Decision Tree Classifier

Classification Report:

	precision	recall	f1-score	support
Defense Evasion	0.97	0.96	0.96	3938
Discovery	1.00	1.00	1.00	46429
Execution	0.99	1.00	0.99	18632
Harmless	0.14	0.18	0.16	441
Impact	0.40	0.22	0.29	9
Other	1.00	1.00	1.00	154
Persistence	1.00	1.00	1.00	42259
micro avg	0.99	0.99	0.99	111862
macro avg	0.78	0.77	0.77	111862
weighted avg	0.99	0.99	0.99	111862
samples avg	0.99	0.99	0.99	111862

Figure 11: Classification Report - Decision Tree Classifier

The classification report for the Decision Tree model shown in Figure 11 demonstrates excellent performance with near-perfect scores in precision and recall for common intents like Discovery, Execution, Other, and Persistence. However, the model struggles with the less frequent intents such as Harmless and Impact, highlighting a potential area for improvement in classifying these rarer and more nuanced categories. Overall, the model achieves high average scores across metrics, indicating robust effectiveness but underscoring the need for targeted enhancements to better address infrequent attack types. This analysis is vital for fine-tuning the model to ensure comprehensive and reliable detection capabilities across a range of attack scenarios.

- **Random Forest Classifier:**
  - **Performance Evaluation:**
    The use of the **Random Forest Classifier** in our project provided compelling results. After training on the dataset, the model achieved a training accuracy of nearly 100% (0.9995) and a testing accuracy

of 97.08%. These high accuracy levels indicate that the Random Forest model is extremely effective at classifying SSH shell attack intents on the training data, suggesting excellent learning of the patterns. However, the slight discrepancy between training and testing accuracies may point towards minor over-fitting, where the model is potentially too tailored to the training data. This difference highlights the need for further tuning to ensure the model generalizes as effectively on unseen data.
Moreover, the Random Forest Classifier demonstrated excellent performance metrics in our analysis: a Hamming Loss of 0.0049 indicates minimal mis-classification; Precision of 0.9919 confirms high accuracy in positive predictions; Recall of 0.9946 shows effective identification of relevant instances; and an F1-score of 0.9932 signifies a strong balance between precision and recall. These results underscore the model's robust capability in accurately classifying SSH shell attack intents.

– **Confusion Matrix:**

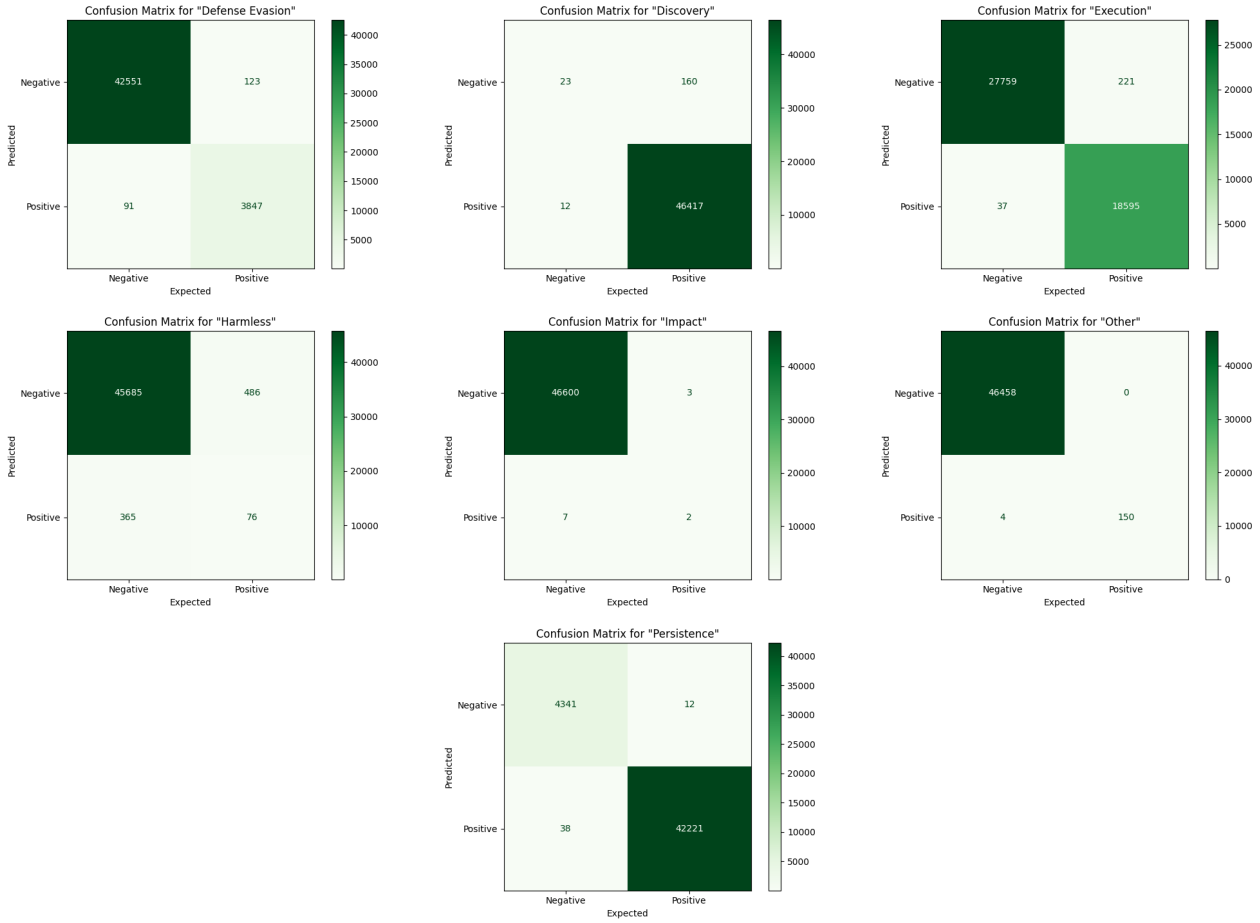


Figure 12: Confusion Matrices - Random Forest Classifier

The Random Forest Classifier demonstrated robust performance across most SSH shell attack intents, with particularly strong predictive accuracy for common categories such as Discovery, Execution, and Persistence. The confusion matrices revealed excellent true positive and true negative counts for these intents. However, the model faced challenges with the rarer intents like Harmless and Impact, which showed lower true positive rates. Additionally, the Defense Evasion category experienced a relatively high number of false positives, suggesting some confusion with other intents. These results indicate that while the Random Forest Classifier is effective for the majority of intents, further tuning is necessary to improve its accuracy for less frequent attack types.

– **Classification Report:**

Classification Report:

	precision	recall	f1-score	support
Defense Evasion	0.97	0.97	0.97	3938
Discovery	1.00	1.00	1.00	46429
Execution	0.99	1.00	0.99	18632
Harmless	0.14	0.18	0.16	441
Impact	0.50	0.22	0.31	9
Other	1.00	0.97	0.99	154
Persistence	1.00	1.00	1.00	42259
micro avg	0.99	0.99	0.99	111862
macro avg	0.80	0.76	0.77	111862
weighted avg	0.99	0.99	0.99	111862
samples avg	0.99	1.00	0.99	111862

Figure 13: Classification Report - Decision Tree Classifier

The Random Forest Classifier demonstrated strong performance across most SSH shell attack intents, achieving nearly perfect precision, recall, and F1-scores for prevalent intents like Defense Evasion, Discovery, Execution, and Persistence. However, it faced challenges with rarer intents such as Harmless and Impact, where both precision and recall were significantly lower. Overall, the classifier's efficiency is highlighted by high overall averages, with macro averages demonstrating the model's capability across a diverse set of intents. This analysis confirms the model's effectiveness in identifying common attack patterns while also indicating areas for improvement in classifying less frequent attack types.

Based on the analysis, the Random Forest Classifier consistently outperforms the Decision Tree model across various SSH shell attack intents. It demonstrates higher precision, recall, and F1-scores, particularly for common intents, and shows slightly better results for rarer intents. Its ability to generalize more effectively and reduced propensity for over-fitting make it the more robust and reliable choice for classifying SSH shell attacks in our multi-label setup. Overall, the Random Forest provides superior performance, making it the preferred model for this application.

2.3 Hyper-parameters tuning of the models through cross-validation

The hyper-parameter tuning is performed for the Decision Tree model within a Binary Relevance framework to optimize its performance for multi-label classification. Utilizing GridSearchCV with a custom scoring function designed for multi-label accuracy, we explored various combinations of hyper-parameters. The optimal configuration found used the 'gini' criterion with no maximum depth, a minimum of 2 samples at leaf nodes, and 5 samples required to split nodes. This configuration led to an impressive cross-validated accuracy of approximately 99.68%. These results demonstrate that with appropriate tuning, the Decision Tree model can effectively handle complex multi-label classification tasks, achieving high accuracy while capturing the nuances of the data.

2.4 Features exploration

The exploration of different features and their combinations revealed that TF-IDF provided the most consistent improvement in model performance. Other features like timestamps and command-specific metrics did not consistently enhance the predictive capabilities of our models. This exploration highlights the importance of feature relevance in model training and suggests that focusing on text-based features, particularly those enhanced by TF-IDF, is most effective for our specific classification tasks involving SSH shell attacks.

Section 3 – Unsupervised learning – clustering

In this section, we explore the application of unsupervised learning to group SSH shell attacks into clusters based on similarities in their textual content. This clustering analysis aims to uncover inherent groupings and patterns within the attack data that may not be immediately obvious through supervised methods. We will employ various clustering algorithms to determine the optimal grouping of the data, supported by techniques such as the elbow method and silhouette analysis for selecting the number of clusters. This analysis will provide a deeper understanding of the attack landscape, revealing how different attacks relate and differ from each other. By examining the characteristics of each cluster, we can gain insights into potential new categories of attacks and better understand the diversity of strategies used by adversaries.

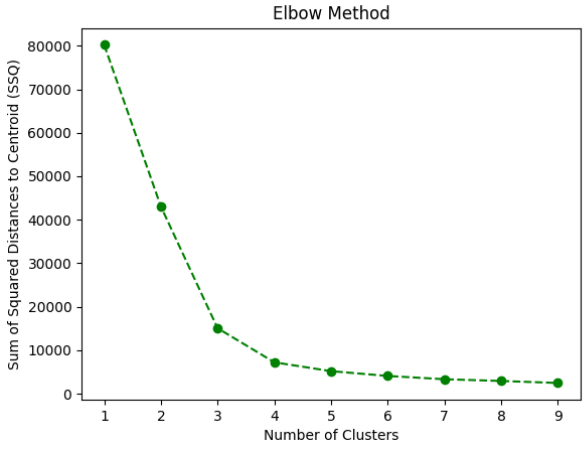
Two cluster algorithms are chosen:

- K-means
- Gaussian Mixture Model (GMM)

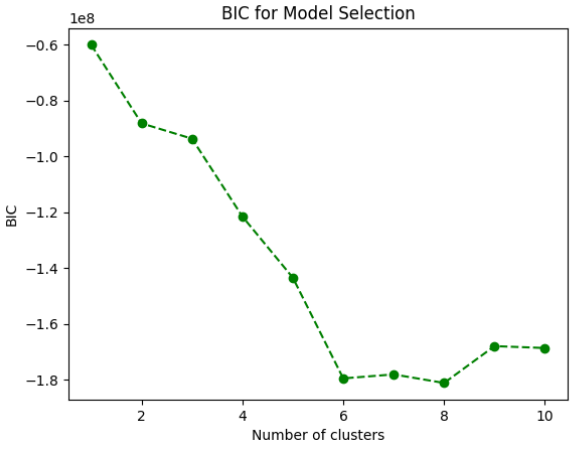
3.1 Determining the number of clusters

In this unsupervised learning task, we aimed to cluster SSH attack sessions based on their word usage to identify patterns and group similar attacks. To determine the optimal number of clusters, we employed two methods: the elbow method and silhouette analysis.
Two methods are used:

- **Elbow Method:** This method helps to decide where adding a cluster is negligible.



(a) Elbow Method Results - K-Means



(b) Elbow Method Results - GMM

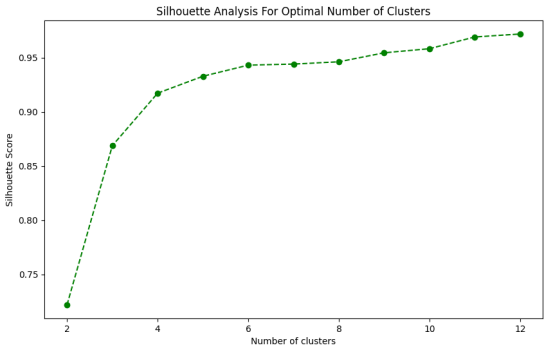
The results show that, for:

  - K-Means: The optimal number of clusters is identified at 4, increasing the number of clusters ceases beyond 4 doesn't yield to significant reductions in the sum of squared distances.
  - GMM: The Bayesian Information Criterion (BIC) suggests that 6 clusters provide the best balance between model complexity and goodness of fit, as indicated by the stabilization of BIC values beyond this point.

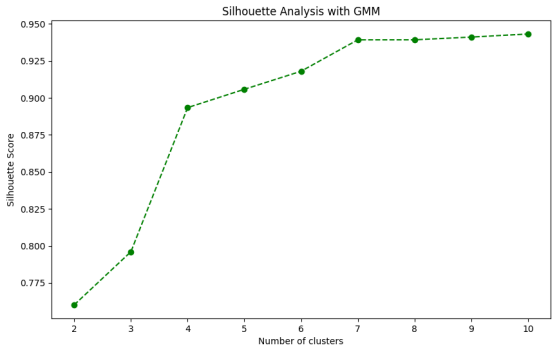
- **Silhouette Analysis:** It is the measure of consistency within clusters of data. Its score ranges from -1 to 1, with higher scores indicating better-defined clusters.

  Due to the **huge number of samples** in our data, the silhouette analysis is too slow. Therefore, it is applied on a sub-sample of all the data (%).

  The results are illustrated in the following figures.



(a) Silhouette Analysis Results - K-Means



(b) Silhouette Analysis Results - GMM

The results show that, for:

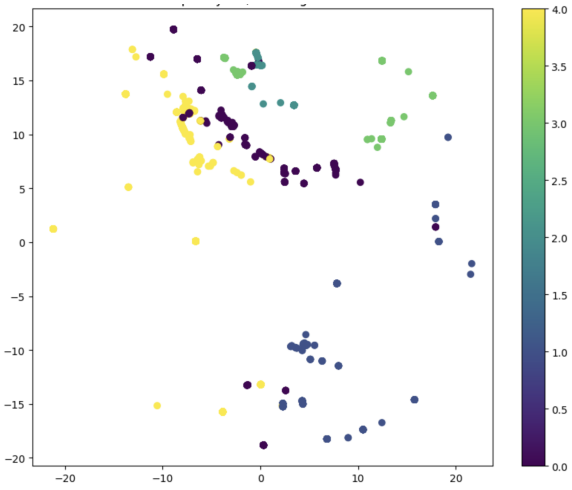
- K-Means: The Silhouette score increased as the number of clusters rose from 2 to 5, after which the improvement plateaued, suggesting that five clusters provide a robust clustering solution without unnecessary complexity.
- GMM: The Silhouette score for increased slowly and reached its highest at seven clusters. This indicates that GMM optimally captured the data patterns with seven clusters, and adding more clusters beyond this point did not enhance the analysis.

3.2 Hyper-parameters tuning

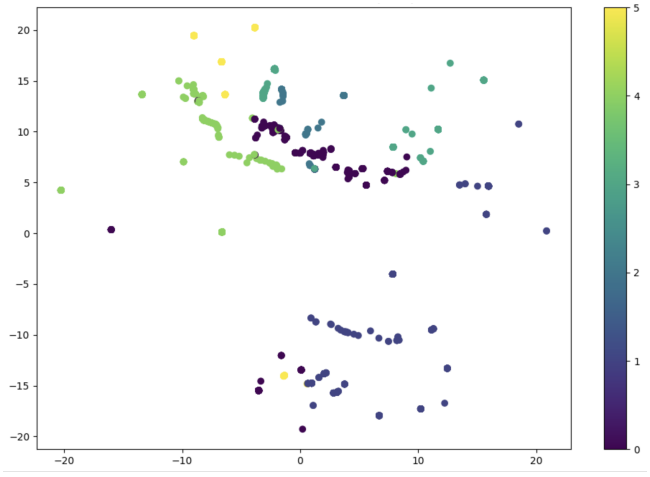
The hyper-parameter tuning is performed to optimize the clustering performance for both K-Means and Gaussian Mixture Model (GMM) algorithms.

3.3 Clusters visualization - t-SNE

Principal Component Analysis (PCA) is employed to decrease the dimensionality of data as a pre-processing step. Then the clusters are visualised through the t-SNE tool. The results are illustrated in the following figures:



(a) t-SNE visualisation - K-Means



(b) t-SNE visualisation - GMM

The results show that, for:

- K-Means: Exhibits clear and distinct cluster boundaries, but with some overlap between clusters. The varied silhouette scores suggest that while some groups are well-defined, others may suffer from ambiguities in membership, reflecting the limitations of K-Means in handling complex data structures.
- GMM: Displays a more fluid distribution of data points with overlapping clusters, indicative of its soft clustering approach which allows for probabilistic cluster membership. This results in less defined boundaries, suggesting a nuanced understanding of the data, potentially capturing more complex patterns.

In summary, K-Means provides sharper segmentation, ideal for simpler, well-separated data, whereas GMM offers a more detailed and blended view, suitable for data with inherent overlaps and mixed distributions. The choice between them should depend on the nature of the dataset and specific analytical needs.

3.4 Clusters analysis

The characteristics of each cluster is analysed and the most frequent words are examined within each cluster using word clouds.

- **K-Means:** Since the optimal number of clusters is around 4, the number of clusters is fixed to 4 for this analysis. The results are illustrated in Figure **??**

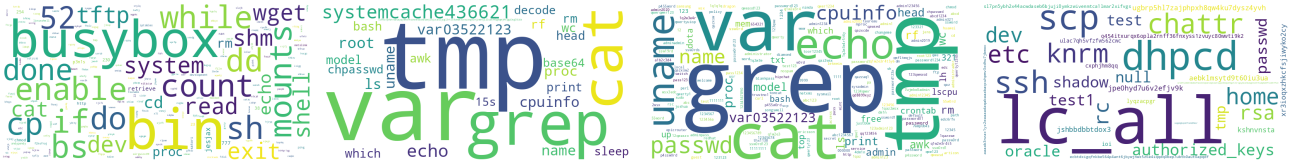


Figure 17: Word cloud - K-Means

  - First Cluster: Frequent use of **busybox**, indicating automated scripts, along with file operations and system modifications.

  - Second Cluster: Focus on **tmp** and **var** directories, data extraction and processing activities.

  - Third Cluster: User and system configuration changes, network-related commands indicating network configuration manipulation.

  - Fourth Cluster: Emphasis on directory navigation and data extraction, system information gathering.

- **GMM:** Since the optimal number of clusters is around 6, the number of clusters is fixed to 6 for this analysis. The results are illustrated in the following figure.

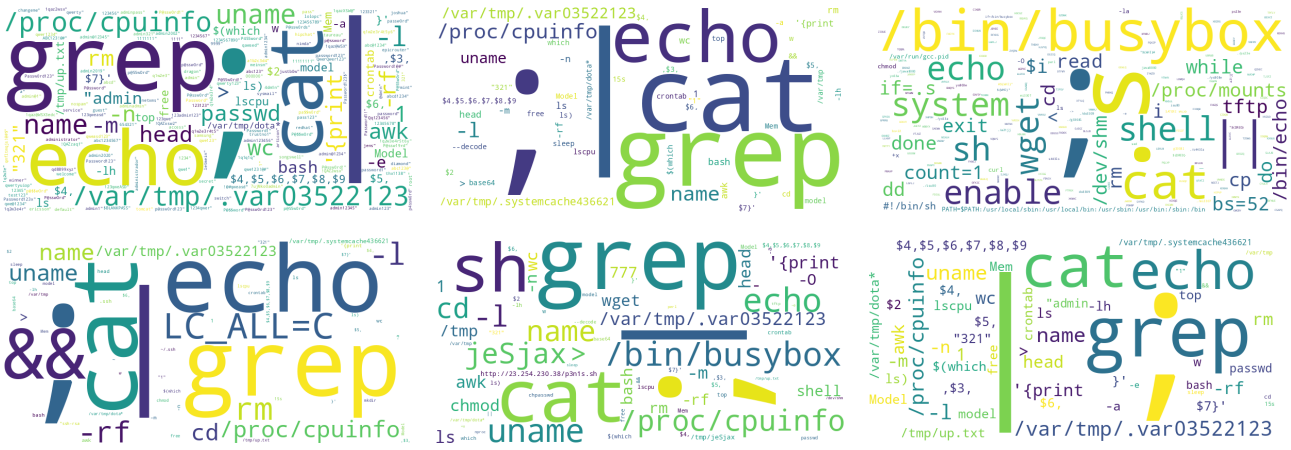


Figure 18: Word cloud - GMM

  - First Cluster: Predominantly uses **uname**, **grep**, and **/proc/cpuinfo**, indicating a focus on gathering system configuration and performance data for reconnaissance purposes.

  - Second Cluster: Characterized by the use of **cat**, **grep**, **var**, and **tmp**, suggesting extensive interactions with temporary and variable storage directories for data extraction and monitoring activities.

  - Third Cluster: Dominated by **busybox**, **sh**, and **echo**, reflecting a strong emphasis on automation and scripting for efficient system setup and modification.

  - Fourth Cluster: Utilizes **rm, cd, echo**, and **grep**, focusing on administrative commands that manage files and system navigation, indicative of system control and administrative tasks.

– Fifth Cluster: Features **ssh, wget**, and **chmod**, pointing to activities related to network access and security configuration, likely aimed at setting up remote operations or securing downloaded tools.

– Sixth Cluster: Involves **top, passwd, cat**, and **grep**, highlighting system monitoring and user management operations, typically for overseeing system performance and enhancing security protocols.

3.5 Intent division reflection

– **K-Means:**

* First Cluster: Discovery, Execution, Impact.
* Second Cluster: Discovery, Defense Evasion, Impact.
* Third Cluster: Persistence, Privilege Escalation, Lateral Movement.
* Fourth Cluster: Discovery, Execution, Credential Access.

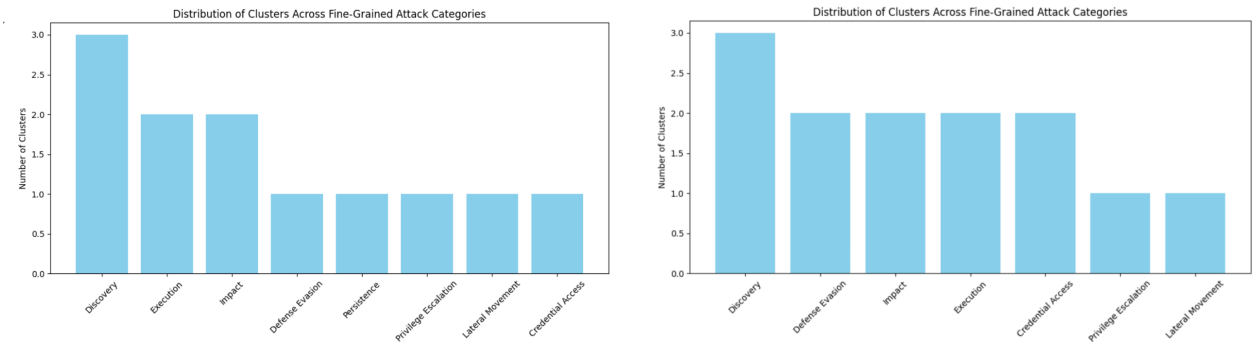
The clusters reflect different stages and tactics of the attack lifecycle, but there is some overlap in intents, indicating that some clusters are heterogeneous in terms of intents.

– **GMM:**

* First Cluster: Discovery.
* Second Cluster: Discovery, Defense Evasion, Impact.
* Third Cluster: Execution.
* Fourth Cluster: Execution, Credential Access, Privilege Escalation.
* Fifth Cluster: Discovery, Credential Access, Lateral Movement.
* Sixth Cluster: Defense Evasion, Impact.

The analysis of the six clusters highlights their alignment with different stages and tactics of the attack lifecycle, showing both homogeneous and mixed intents. Some clusters distinctly focus on specific tasks like system reconnaissance or execution, while others blend multiple tactics, such as combining discovery with impact or lateral movement. This variation underscores the complexity of cybersecurity threats, where similar command sets can serve overlapping and diverse objectives in an attack scenario.

3.6 Clusters of similar attacks



(a) Clusters of similar attacks - K-Means

(b) Clusters of similar attacks - GMM

The results show that for:

– K-Means: Shows a predominant cluster in **Discovery** suggesting a common or dominant attack pattern easily identifiable by this method. Other categories are represented less frequently, indicating that K-Means might struggle with more nuanced behaviors that do not form clear spherical clusters.

– GMM: Demonstrates a more balanced distribution across categories, with "Discovery" having the most clusters. This indicates GMM's strength in capturing a wider variety of attack behaviors due to its flexibility in handling clusters of different shapes and densities.

Section 4 – Language Models exploration

This section investigates the potential of language models to enhance the classification of SSH shell attacks. Language models, particularly those pre-trained on large datasets, offer advanced capabilities for encoding textual data into meaningful representations. We will compare the effectiveness of different language models, such as BERT and Doc2Vec, in capturing the nuances of SSH session texts. The focus will be on fine-tuning these models to adapt to our specific cybersecurity context, thereby improving our ability to discern and classify attack intents. We will examine the training dynamics, including learning curves and optimal stopping points, to optimize model performance. This exploration into language models seeks to leverage state-of-the-art NLP technologies to advance the field of cybersecurity analytics.

- **Language model selection: Bert** language model is selected for the NLP tasks for the following reasons:
  - **Pre-training:** BERT is pre-trained on extensive corpora, including Wikipedia and BookCorpus, providing rich, generalizable text representations.
  - **Task Adaptability:** BERT can be fine-tuned for specific tasks like text classification and named entity recognition with minimal additional layers, outperforming Doc2Vec, which requires more effort and doesn't achieve the same performance level.
  - **Ease of Use:** Supported by the Hugging Face Transformers library, BERT offers user-friendly implementations and pre-trained models, simplifying our task with ample examples and resources.
- **Data preparation:** A dataset of 233,035 rows and 4 columns is loaded, initially testing with a smaller sample for repeatability. Focusing on 'full session' and 'set fingerprint' columns, created binary columns for each of the seven intents, and generated a new 'labels' column with binary values for each session's intents. The dataset was then split into training (80%) and test (20%) sets.
  Training on 230,000 rows is computationally intensive and time-consuming, especially for complex models like BERT. Therefore, the stratified sampling to reduce the training size data (around 45 %).
  Then, the data was prepared for BERT by adding special tokens, padding/truncating sentences, creating attention masks, tokenizing with the BERT tokenizer, and using torch DataLoader for efficient memory usage during training.
- **Model architecture and training**
  - **Customizing BERT for Classification:** The pre-trained BERT model was customized for multi-label classification using a custom class and the BCEWithLogitsLoss function, which combines a sigmoid layer with binary cross-entropy loss.
  - **Fine-tuning the Last Layer of the Network**

    To fine-tune the last layer of the network on the supervised training set, we initially set the training epochs to N=4 for a sample of 100,000. However, this proved to be excessive. Consequently, for the entire dataset, we adjusted the training epochs to N=3. This adjustment optimizes the balance between computational efficiency and model performance.
  - **Training the Model:** The BERT model's performance was evaluated on different sample sizes and epochs.
    The learning curves show a positive trend in model performance, with validation accuracy consistently improving to approximately 0.994 and the validation F1 score remaining stable at around 0.994, indicating strong performance across multiple classes without significant over-fitting. The training loss decreases from approximately 0.03 to 0.0125, while the validation loss drops from around 0.015 to below 0.0125, demonstrating the model's effective learning and improving generalization capability on unseen data.

    In summary, the model shows consistent improvements across epochs, with optimal training typically achieved within 2-3 epochs. This minimizes unnecessary computation and over-fitting while maintaining high performance. For instance, with the full dataset over three epochs, training loss decreased from 0.03 to 0.0125, validation loss from 0.015 to below 0.0125, validation accuracy from 0.980 to 0.994, and F1 score from 0.994 to 0.994, confirming the model's effectiveness and stability.

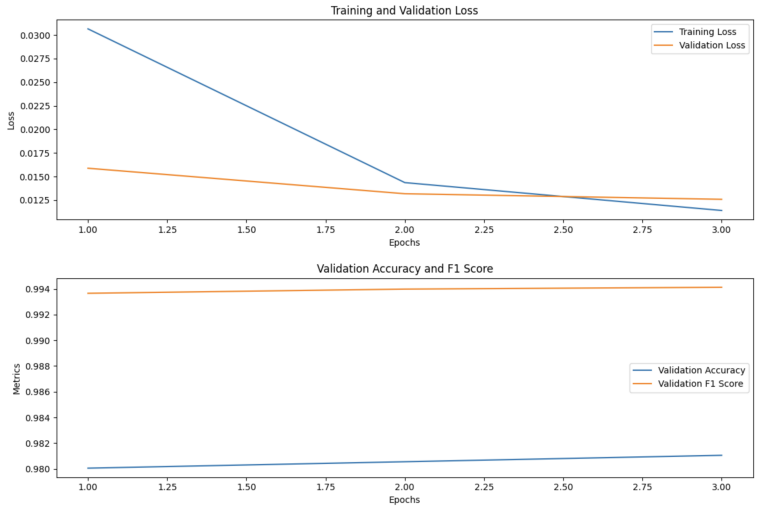


Figure 20: Performance metrics vs. epochs

- **Interpretation of the results:** The BERT model showed significant improvement in the first two epochs, with diminishing returns in the third. Training loss and validation loss consistently decreased, while validation accuracy and F1 score remained high and stable, indicating effective learning, good generalization, and minimal over- fitting. This demonstrates BERT's suitability for classifying SSH attack sessions.