SSH Shell Attacks - Appendix

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

Contents

A DATA EXPLORATION AND PRE-PROCESSING

This appendix contains additional plots and visualizations related to the data exploration and pre-processing phase of the analysis. The figures are grouped into subsections based on their thematic relevance.

A.1 Temporal Analysis of Attacks

This subsection presents visualizations related to the temporal distribution of SSH attacks, including trends over hours, days, months, and years.

A.1.1 Attack Frequency by Hour

The plot shows the distribution of SSH attacks across different hours of the day. It reveals specific hours when attack activity peaks, which could indicate targeted times for malicious activities. Understanding these patterns can help in implementing time-based security measures.

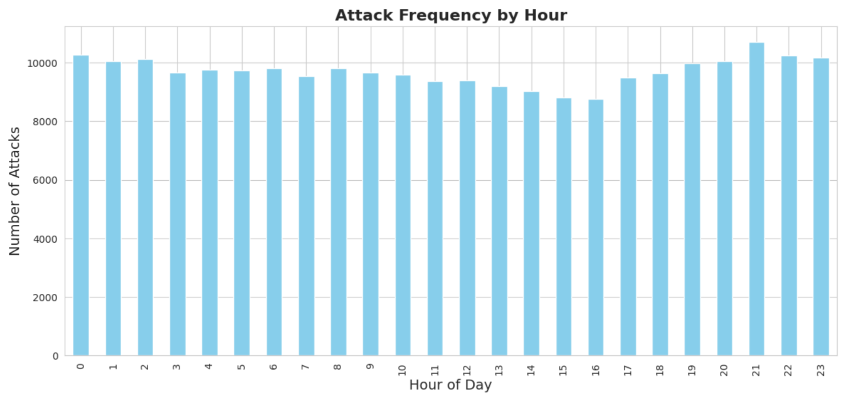

Fig. 1. Distribution of SSH attacks by hour of the day. The plot highlights peak hours during which attacks are most frequent.

*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.

A.1.2 Attack Frequency by Month

This plot illustrates the distribution of SSH attacks across different months. It highlights seasonal trends, showing months with higher attack frequencies. Such insights can be useful for anticipating periods of increased security threats and allocating resources accordingly.

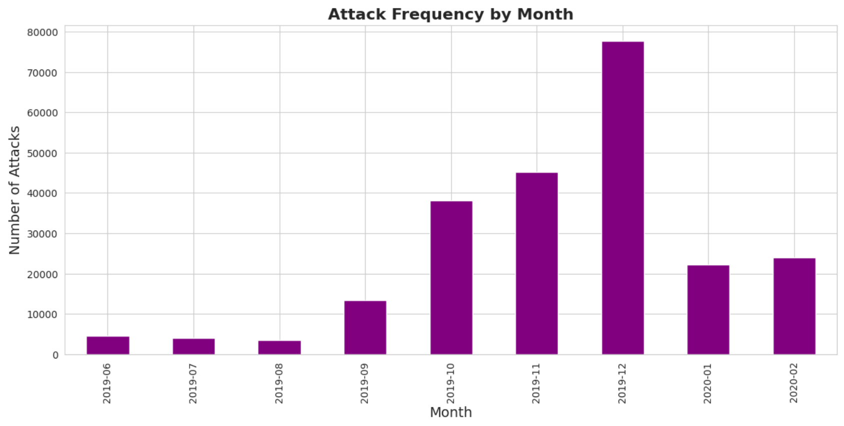

Fig. 2. Distribution of SSH attacks by month. The plot reveals seasonal trends in attack frequency.

A.1.3 Attack Frequency by Year

The plot compares the frequency of SSH attacks between the years 2019 and 2020. It provides a clear view of how attack patterns have evolved over time, indicating whether there has been an increase or decrease in malicious activities. This information is crucial for understanding long-term trends and evaluating the effectiveness of security measures.

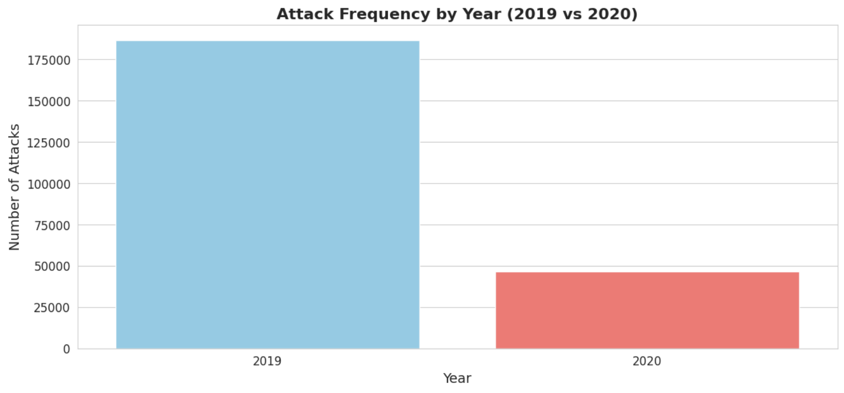

Fig. 3. Distribution of SSH attacks by year. The plot shows the overall trend of attacks over multiple years.

A.1.4 Temporal Series of SSH Attacks

This time series plot provides a detailed view of SSH attack patterns over the entire dataset. It captures fluctuations in attack frequency, highlighting periods of heightened activity. Such a comprehensive view is essential for identifying anomalies and understanding the overall dynamics of SSH attacks.

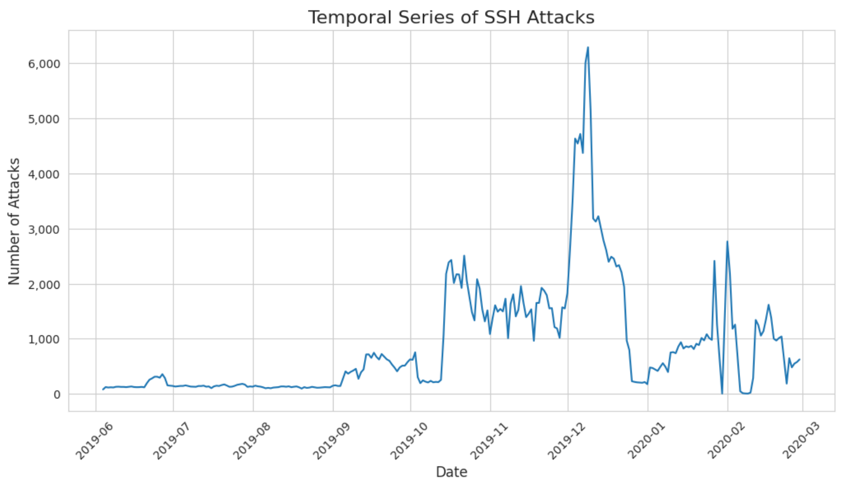

Fig. 4. Time series plot of SSH attacks over the entire dataset. The plot provides a comprehensive view of attack patterns over time.

A.2 Intents Over Timestamps

This subsection explores the relationship between attack intents and their timestamps.

A.2.1 Intents Over Timestamps

The plot visualizes the distribution of different attack intents over time. It categorizes intents such as Defense Evasion, Harmless, Impact, Discovery, Persistence, Execution, and Others. By analyzing this plot, we can identify which intents are more prevalent at specific times, helping to prioritize security measures based on the nature of the threats.

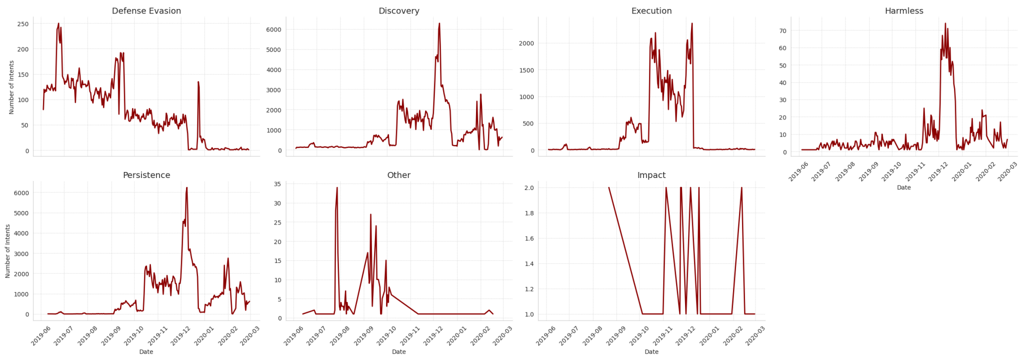

Fig. 5. Visualization of attack intents over timestamps. The plot provides insights into the temporal patterns of different attack intents.

B SUPERVISED LEARNING - CLASSIFICATION

B.1 Logistic Regression

In this section, we detail the steps and results obtained from using the Logistic Regression model during the supervised learning phase of the project. Logistic Regression served as a baseline model to provide an initial understanding of the classification problem. Despite its simplicity, it offered valuable insights into the multi-label classification task.

B.1.1 Model Training

The Logistic Regression model was trained using its default configuration. Specifically, the `lbfgs` solver was utilized with a regularization parameter $C = 1$. The training process aimed to identify potential overfitting or underfitting issues and establish baseline performance metrics.

```
# Initialize and train Logistic Regression model
model = LogisticRegression(max_iter=1000, random_state=42)
model.fit(X_train_tfidf, y_train_binary)
```

Listing 1. Train Logistic Regression model

The dataset was preprocessed using the TF-IDF representation of the session texts, which assigned weights to words based on their frequency and relevance within the dataset. Multi-label binary encoding was applied to the ‘Set_Fingerprint‘ column to ensure compatibility with the model.

B.1.2 Evaluation Metrics

The Logistic Regression model was evaluated using standard classification metrics, including weighted F1-scores, precision, and recall. The evaluation metrics highlighted the strengths and weaknesses of the model in handling imbalanced classes.

```
# Generate classification report
from sklearn.metrics import classification_report
report = classification_report(y_test_binary, y_pred, zero_division=0)
print(report)
```

Listing 2. Generate classification report

The confusion matrix provided a breakdown of true positives, false positives, false negatives, and true negatives for each intent. Figure 6 shows the confusion matrix for the Logistic Regression model.

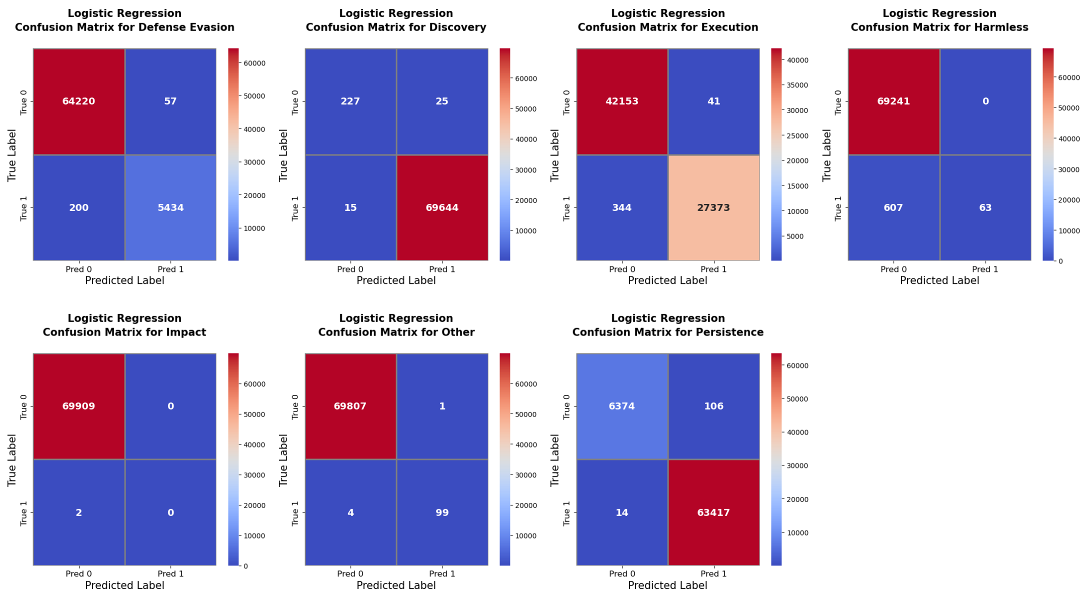

Fig. 6. Confusion Matrix for Logistic Regression Model.

B.1.3 Hyperparameter Tuning

Grid search was performed to optimize the Logistic Regression model's hyperparameters. The search focused on varying the regularization parameter C over a range of values [0.1, 1, 10, 100] to identify the configuration that maximized weighted F1-scores.

```
# Define parameter grid for Logistic Regression
param_grid = {'C': [0.1, 1, 10, 100]}
grid_search = GridSearchCV(LogisticRegression(max_iter=1000,
    random_state=42), param_grid, cv=5)
grid_search.fit(X_train_tfidf, y_train_binary)
```

Listing 3. Parameter grid for Logistic Regression

The optimized model exhibited improved performance compared to the baseline, particularly for intents with smaller sample sizes. Figure 7 illustrates the weighted F1-scores for different values of C.

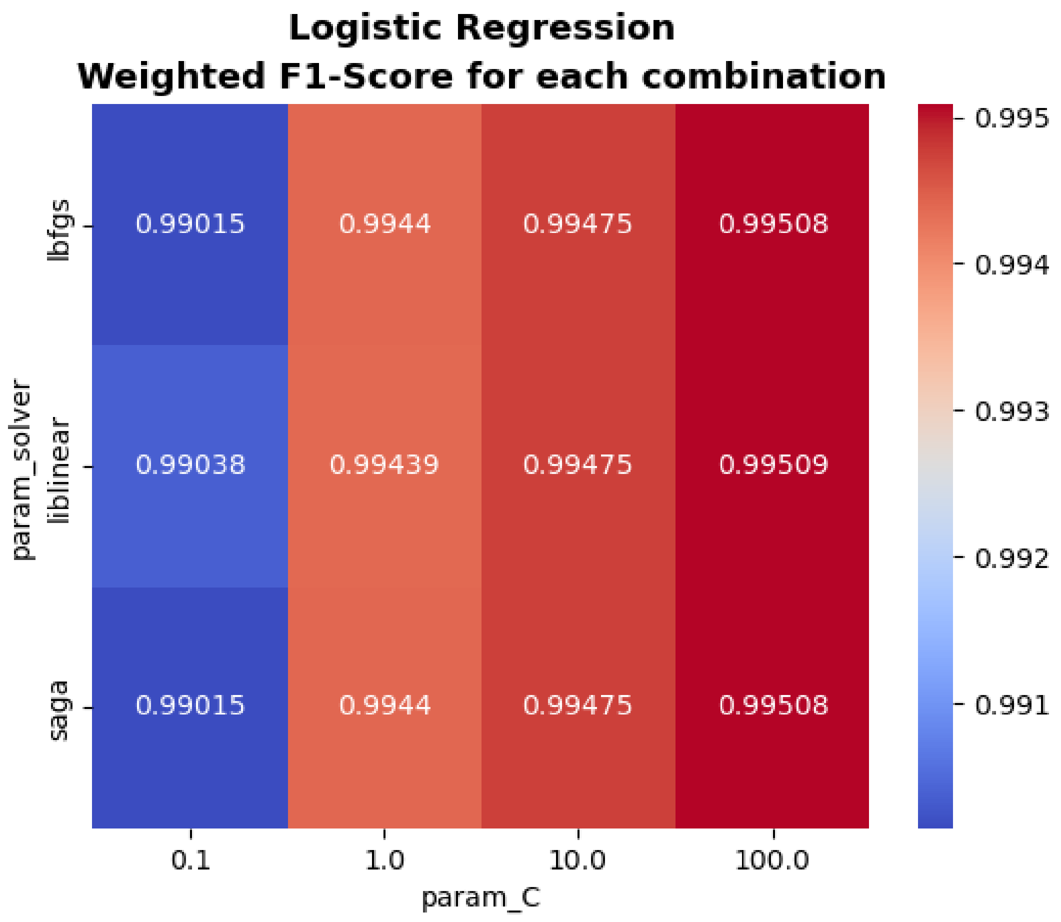

Fig. 7. Weighted F1-Scores for Logistic Regression Hyperparameter Tuning.

B.1.4 Comparative Analysis of Baseline and Optimized Models

The optimized Logistic Regression model demonstrated a moderate improvement in precision and recall compared to the baseline. However, its overall performance remained slightly inferior to more complex models like Random Forest and SVM. The comparative analysis underscores the importance of selecting models suited to the dataset's characteristics and problem requirements.

C UNSUPERVISED LEARNING - CLUSTERING

...

D LANGUAGE MODEL EXPLORATION

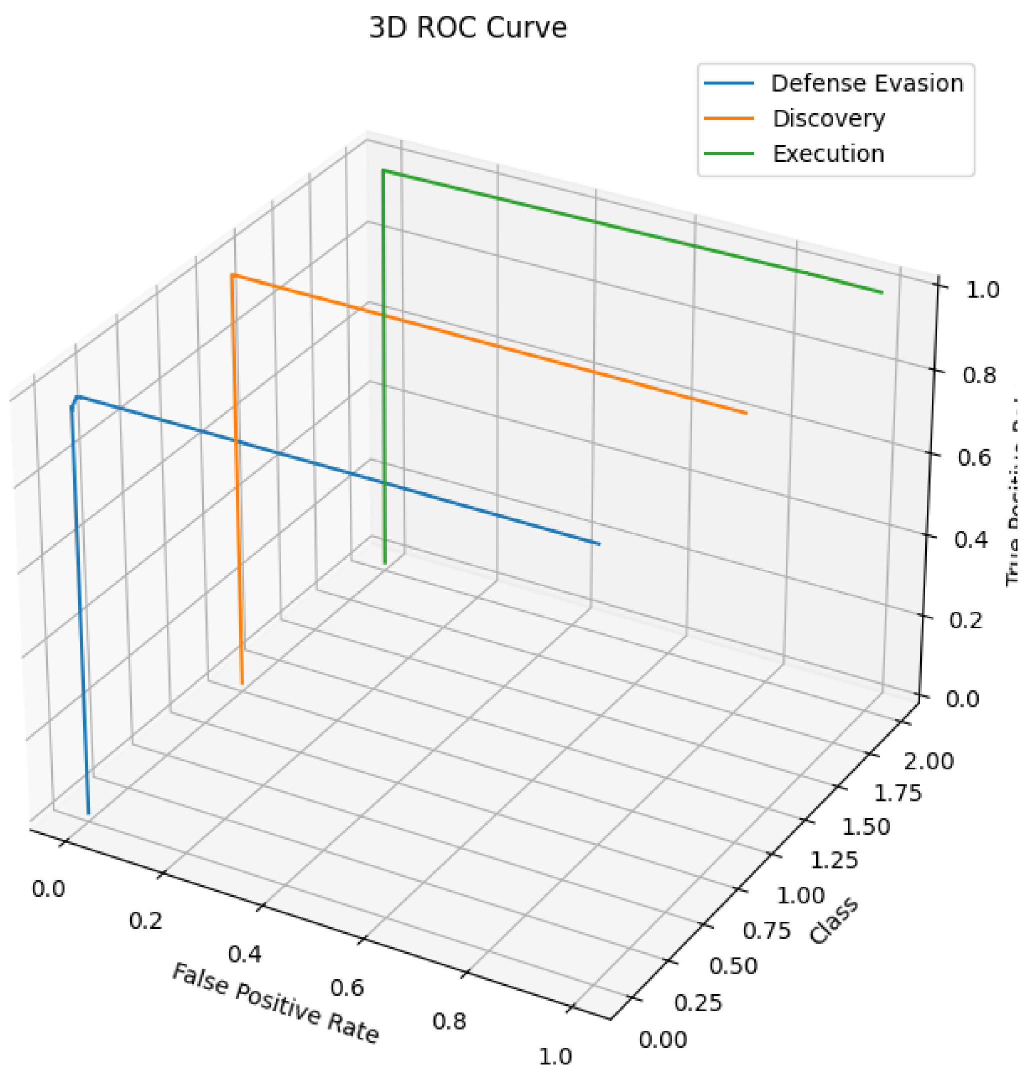

Fig. 8

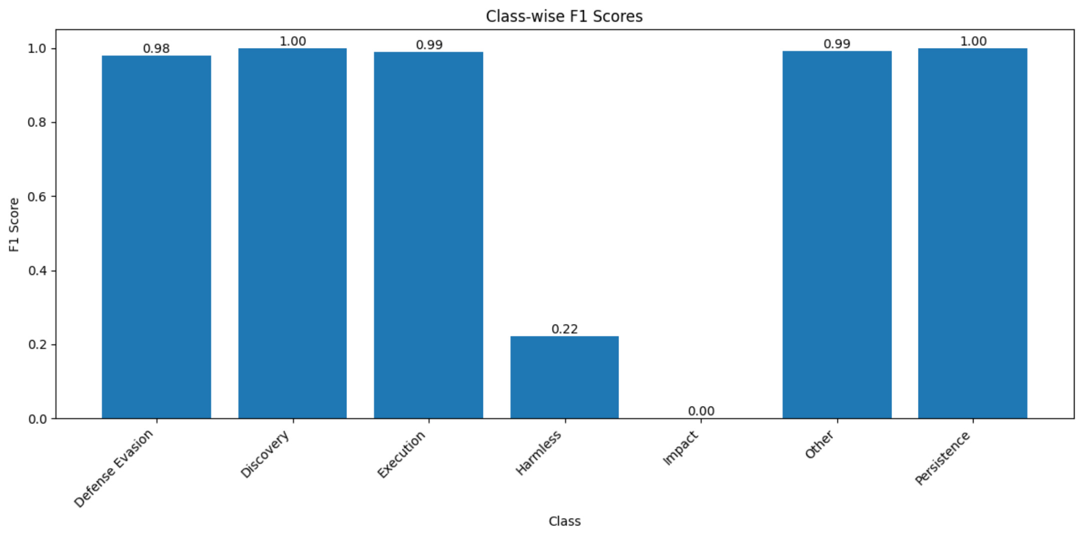
Class-wise F1 Scores
F1 Score
Class
0.98
1.00
0.99
0.22
0.00
0.99
1.00
Defense Evasion
Discovery
Execution
Harmless
Impact
Other
Persistence

Fig. 9

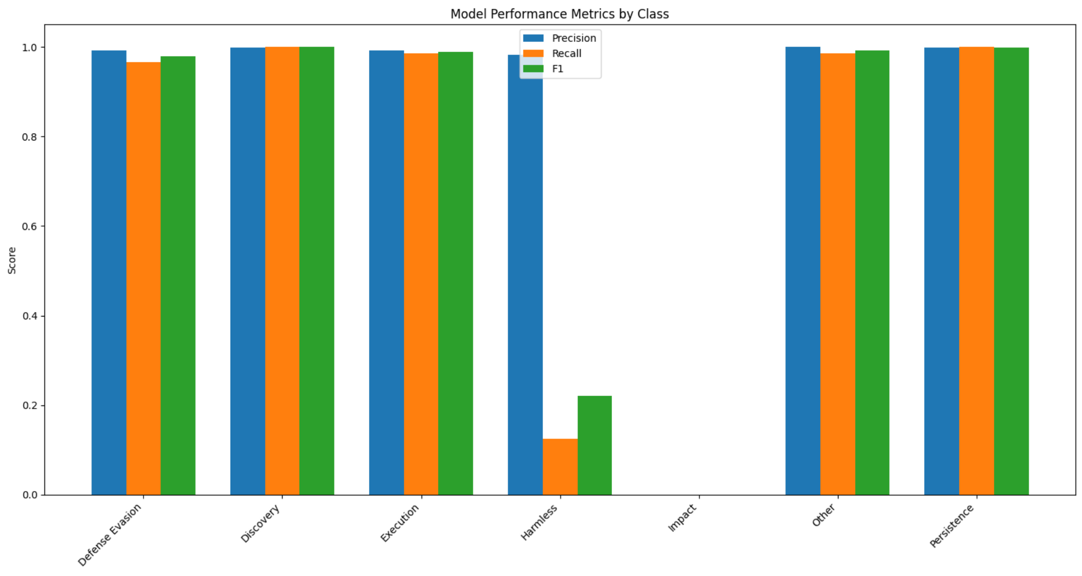
Model Performance Metrics by Class
Precision
Recall
F1
Score
Defense Evasion
Discovery
Execution
Harmless
Impact
Other
Persistence

Fig. 10

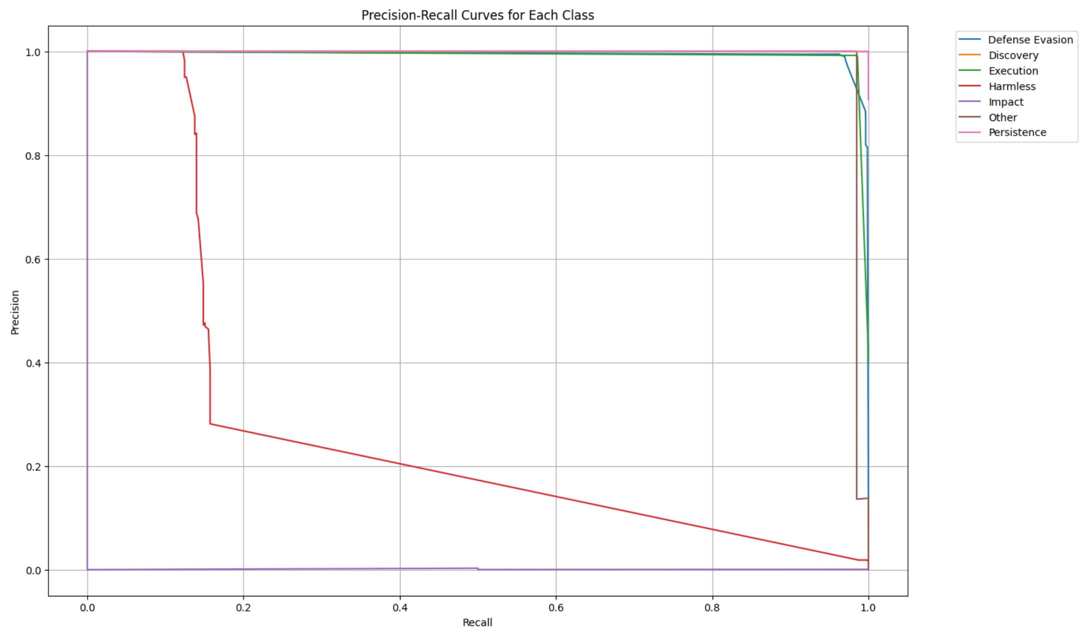
Precision-Recall Curves for Each Class
Precision
Recall
0.0
0.2
0.4
0.6
0.8
1.0
Defense Evasion
Discovery
Execution
Harmless
Impact
Other
Persistence

Fig. 11

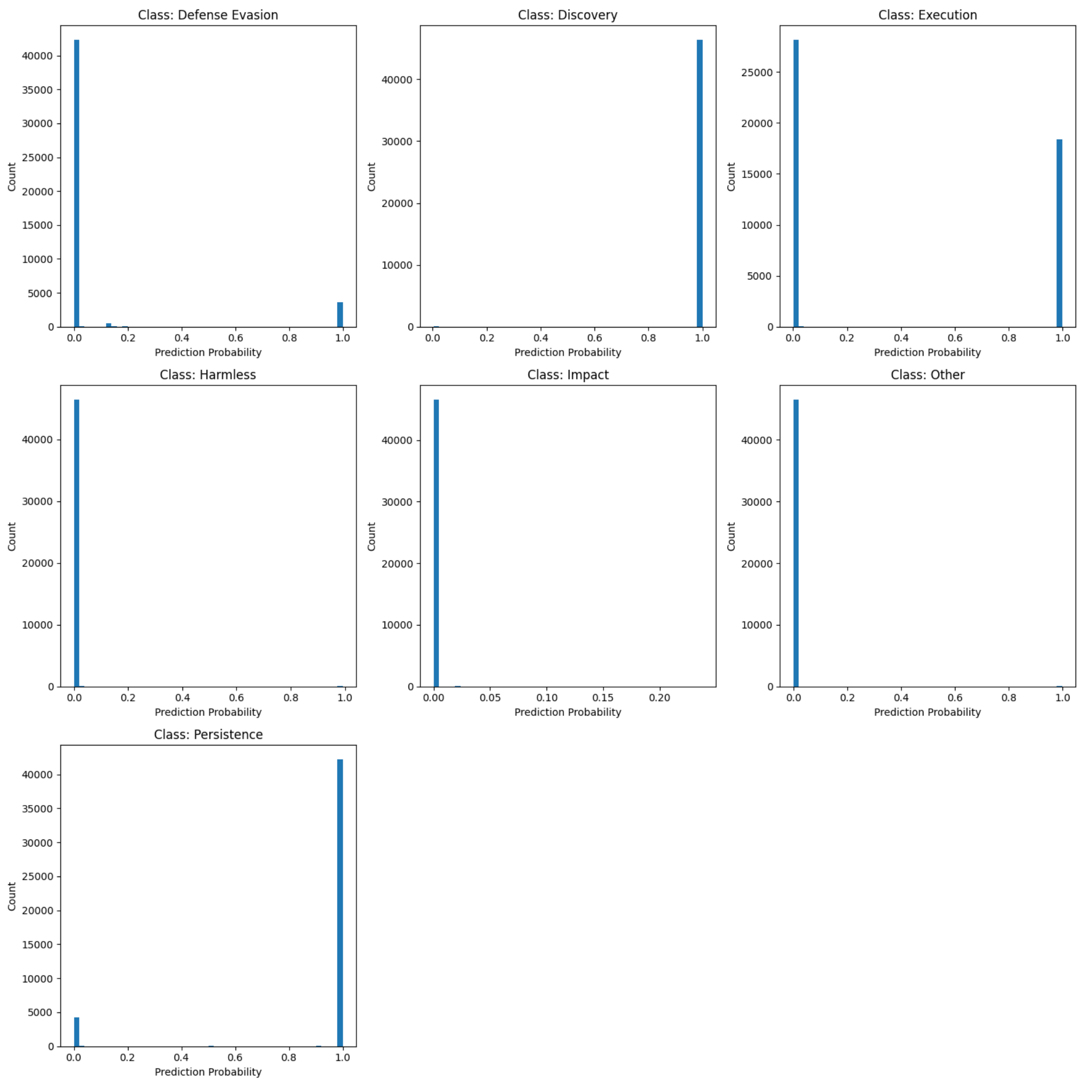
Class: Defense Evasion
Class: Discovery
Class: Execution
Class: Harmless
Class: Impact
Class: Other
Class: Persistence
Count
Prediction Probability

Fig. 12