Supplementary File

This document includes the supplemental materials for the paper titled "DEEPDRAC: Disposition Recommendation for Alert Clusters Based on Security Event Patterns".

APPENDIX A
THE QUALITY OF PATTERN CLUSTERS

A high-quality pattern cluster should consist of security events that share the same pattern. A necessary condition for this is that all events within the cluster exhibit a single risk level. Due to the lack of event-level annotations, we adopt a relaxed criterion and define high-quality clusters as pattern clusters that contain only a single risk level of security events. To assess the overall quality of pattern clustering in DEEPDRAC, we measure the proportion of such high-quality clusters and the proportion of alerts they cover.

On the DARPA'99 dataset, 89.57% of the pattern clusters generated by DEEPDRAC contain a single risk level, and the alerts within these clusters account for 97.47% of all alerts. On the EIPGC dataset, 77.26% of the clusters contain only one risk level, while 89.53% contain two adjacent risk levels (e.g., low and medium, or medium and high), covering 73.17% and 91.39% of all alerts, respectively. On the CIC-IDS2017 dataset, 97.37% of the pattern clusters include only one risk level, and the alerts within them account for 98.53% of the total.

Overall, except for the more complex real-world dataset EIPGC, high-quality pattern clusters generated by DEEPDRAC cover over 97% of alerts in the other two datasets. Even for EIPGC, clusters with slightly lower quality (those containing adjacent risk levels) still cover more than 91% of the alerts. These results indicate that the vast majority of the pattern clusters constructed by DEEPDRAC are valid and meaningful.

APPENDIX B
PRACTICAL IMPLICATIONS FOR SOC OPERATORS

In the manual analysis stage, DEEPDRAC significantly reduces the workload for SOC operators by extracting patterns from security events and grouping these patterns into clusters, followed by performing random sampling within each pattern cluster for manual analysis. This approach enables operators to focus on a small, sampled subset of security events from each cluster, rather than exhaustively reviewing all isolated alerts. For instance, on the DARPA'99 dataset, DEEPDRAC reduces the number of alerts requiring manual intervention to just 1.22% of the total, meaning that only 122 sampled security events need to be analyzed out of 10,000 raw alerts. The clustering process provides richer contextual information, such as temporal and spatial correlations among alerts, which enhances the accuracy and reliability of the analysis. By focusing on representative samples, SOC teams can prioritize high-risk events, improve decision-making precision, and streamline their operations. This process eliminates the need for operators to manually correlate raw alert contexts and reduces the analysis workload through sampling-based batch disposition.

In the automated analysis stage, DEEPDRAC further enhances efficiency by leveraging its high accuracy and recall to automatically recommend alert dispositions. For example, on the DARPA'99 dataset, DEEPDRAC achieves an accuracy of 97.96% and a recall of 97.39%, ensuring that the majority of security events are correctly categorized and matched with appropriate responses. Additionally, the low underestimation rate minimizes the likelihood of critical security events being overlooked. By integrating these strengths, DEEPDRAC reduces the cognitive burden on SOC operators and accelerates event response, enabling SOC teams to efficiently dispose of large volumes of alerts while maintaining high accuracy.

APPENDIX C
TIME EFFICIENCY ANALYSIS

To evaluate the practical feasibility of DEEPDRAC in real-world scenarios, this section presents an analysis of its runtime performance and scalability when dealing with alerts of different scales. We use the CIC-IDS2017 dataset, which contains a diverse range of attack categories, as the source of alert data. To assess time consumption, we evaluate both the model inference and training stages across varying alert scales, with $N_{alert} = 1 \sim 500K$.

1) Model Inference: The model inference stage corresponds to the real-world usage of DEEPDRAC by operators in SOC, which consists of two main steps: 1) Reconstructing security events from raw alerts using the enhanced Louvain; 2) Generating recommended disposition results based on GNN embeddings and clustering. Since DEEPDRAC uses a time window of $T = 10$ minutes to correlating and dispose of alerts, we evaluate the time consumption when different alert scales are entered into a single time window. After chronologically sorting the $685.56K$ alerts generated over 5 days from the CIC-IDS2017 dataset, we simulate the rapid increase in alert scale in a short time by feeding $N_{alert} = 1 \sim 500K$ alerts into one time window.

The theoretical time complexity of the Enhanced Louvain algorithm is $O(N \log N + k'E)$, where N is the number of nodes, E is the number of edges, and k' is a constant factor related to the number of frequent itemsets, resulting in an overall near-linear complexity. The detailed analysis is as follows:

1) The Louvain algorithm is applied for the initial community detection with a time complexity of $O(N \log N)$.
2) Each edge within the communities is traversed to split the initial partition into single-center subgraphs with

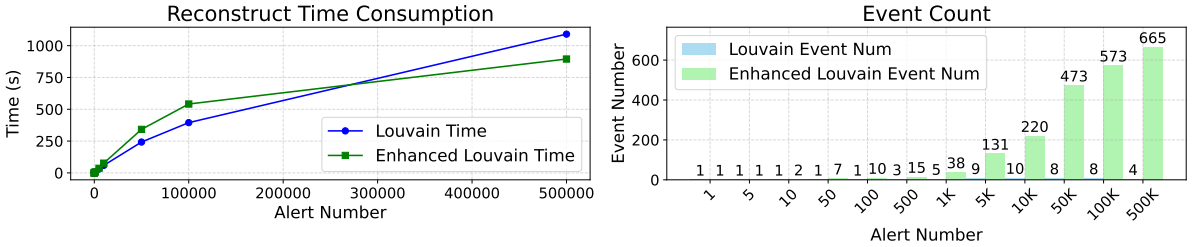

Fig. 9: The runtime performance of DEEPDRAC.

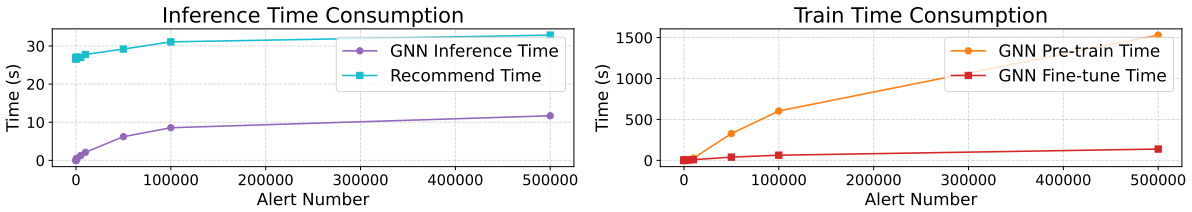

Fig. 10: The runtime performance of DEEPDRAC.

uniform alert categories, while constructing dictionaries keyed by center IP and alert category, and by alert category alone. This step has a time complexity of $O(E)$.

3) The FP-growth algorithm is used to mine frequent itemsets from community transactions, requiring two passes over the transaction set, with time complexity approximately $O(2E)$.
4) Sub-communities with the same center and alert category are merged based on the constructed dictionaries, which takes constant time $O(1)$.
5) Frequent itemsets (assumed to be k in number) are iterated over to merge communities based on the dictionaries, with connectivity checks performed by traversing relevant edges, resulting in a time complexity of $O(kE)$.
6) Communities are traversed to identify and merge two-node subgraphs connected to other communities, with a time complexity of $O(2E)$.

Therefore, the overall time complexity of the Enhanced Louvain algorithm is $O(N \log N + k'E)$, demonstrating efficient, near-linear scalability for large-scale alert graphs. This theoretical analysis is also validated by experimental results. The runtime performance and scalability of reconstructing security events from raw alerts are presented in Fig. 9. Specifically, the time consumption of the enhanced Louvain algorithm designed in DEEPDRAC increases linearly with the number of alerts. Furthermore, comparative experiments with the Louvain algorithm revealed that when faced with large-scale alerts, the Louvain algorithm tends to couple multiple irrelevant security incidents together, generating meaningless large subgraphs. For example, when processing $500K$ alerts (approximately four days of alert data), the Louvain algorithm produced only four subgraphs during partitioning. In contrast, the enhanced Louvain algorithm considers additional knowledge such as alert attributes, leading to a more reasonable decoupling granularity. The process of reconstructing security events involves feature extraction (such as three-node motifs), and the large subgraphs generated by the standard Louvain algorithm under large-scale alerts lead to increased processing time. As a result, at $N_{alert} = 500K$, the reconstruction time using Louvain even exceeds that of the enhanced Louvain.

After reconstructing security events, DEEPDRAC applies GNN inference to obtain embeddings of the events and employs hierarchical clustering to recommend appropriate disposition strategies. As shown on the left side of Fig. 10, the time consumed by GNN inference and hierarchical clustering increases with the number of alerts. Although there is a noticeable rise before $100K$ alerts, the growth flattens significantly beyond that point, indicating that the overall inference time remains manageable even under large-scale alert scenarios. Even when faced with $500K$ alerts, DEEPDRAC can still complete the inference within one minute.

2) Model Training: If all alerts are treated as belonging to a single time window during GNN training, even with $500K$ alerts, the number of reconstructed security events remains only 665. Since GNN training is conducted at the event level, the time consumed is only 58.60 seconds. To better evaluate the time consumption of model training in real-world scenarios, we still follow the original training strategy of DEEPDRAC to partition the data into multiple time windows from 1 to $500K$ according to time intervals to generate historical data for model training.

The time consumption for GNN pre-training and fine-tuning as the number of alerts increases is shown on the right side of Fig. 10. Benefiting from the design of the GNN as a lightweight model (with only 71,719 parameters), even when trained with 500K alerts, the GNN can complete training on a single NVIDIA GeForce RTX 3090 GPU in 27.8 minutes (compared to 35 minutes for DeepCASE). This indicates that DEEPDRAC has low training resource requirements, allowing SOC with a large scale of historical alerts to complete model training quickly and deploy it efficiently.

In summary, DEEPDRAC achieves efficient training on large-scale alerts while maintaining near-linear time complexity during inference. Compared to the standard Louvain

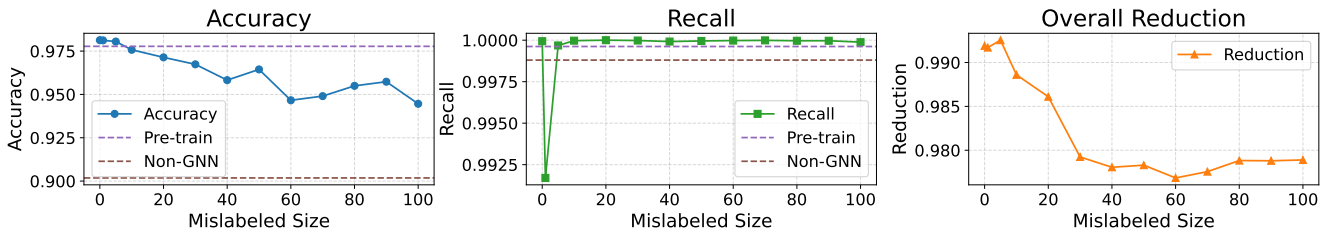

Fig. 11: Performance of DEEPDRAC under different sizes of mislabeled data. The average results over five runs.

algorithm, it offers better scalability under high alert volumes. If the alert scale further increases, alerts can be processed in smaller batches by reducing the time window T to ensure timely response.

APPENDIX D
DETECTION PERFORMANCE ACROSS ATTACK CATEGORIES

TABLE VI: Performance of DEEPDRAC across Different Attack Categories

Attack category	Attacks	TP	TN	FP	FN	Recall	FPR
Bot	240	77	169726	1538	163	32.08%	0.90%
PortScan	36	36	171410	58	0	**100.0%**	**0.03%**
DDoS	32827	32827	31059	107618	0	**100.0%**	77.60%
SSH-Patator	892	892	170612	0	0	**100.0%**	**0.00%**
DoS Hulk	104781	91684	25509	22774	13097	87.50%	47.17%
Web-Brute Force	482	482	152582	0	0	**100.0%**	**0.00%**
Web-XSS	2	0	171502	0	2	0.00%	0.00%
DoS slowloris	586	584	170625	293	2	**99.66%**	**0.02%**
Infiltration	3	2	153061	0	1	66.67%	0.00%
DoS Slowhttptest	796	493	170708	0	303	61.94%	0.00%
DoS GoldenEye	3	0	153061	0	3	0.00%	0.00%
FTP-Patator	1	1	153063	0	0	**100.0%**	**0.00%**

TABLE VII: False Negative and False Positive Analysis of Underperforming Attack Categories

Attack category	Predicted Category of FN	FN Count	Source Category of FP	FP Count
DDoS	/	0	DoS Hulk BENIGN	104772 2846
DoS Hulk	DDoS	13097	DDoS PortScan DoS slowloris	22769 4 1
DoS Slowhttptest	DoS slowloris PortScan	293 10	/	0
DoS GoldenEye	PortScan	2	/	0
Bot	BENIGN	163	BENIGN	1538
Web-XSS	Web-Brute Force	2	/	0
Infiltration	BENIGN	1	/	0

Note: "BENIGN" refers to false positive alerts.

This Appendix includes the performance of DEEPDRAC in different attack scenarios in the CIC-IDS2017 dataset (Table VI), as well as the performance of false positive sources and types of missed predictions for the seven attacks categories with a low score (Table VII). Because the density distribution of GNN embeddings varies across different attack scenarios, we perform clustering and recommendation using two values of eps: 0.02 (as used in Section IV-B) and 0.2.

The False Positive Rate (FPR) refers to the proportion of samples that do not belong to a given attack category but are incorrectly predicted as that category:

$$\text{FPR}_i = \frac{\text{False Positives}_i}{\text{False Positives}_i + \text{True Negatives}_i}$$

where False Positives_i denotes the number of alerts that do not belong to category i but are incorrectly predicted as i, and True Negatives_i denotes the number of alerts that do not belong to category i and are correctly predicted as not i.

APPENDIX E
ROBUSTNESS AND STABILITY OF THE MODEL

This Appendix presents the performance of DEEPDRAC under varying proportions of mislabeled data, including its impact on alert reduction, recommendation accuracy, and recommendation recall. The results are illustrated in Fig. 11.