DYNAMO: Towards Network Attack Campaign Attribution via Density-Aware Active Learning

Helene Orsini[1][a] and Yufei Han[2][b]
[1]*Inria, Univ. Rennes, IRISA, Rennes, France*
[2]*CentraleSupelec, Univ. Rennes, IRISA, Rennes, France*

Keywords: Campaign Attribution, Unseen Campaign Detection, Density-Aware Active Learning.

Abstract: Network attack attribution is crucial for identifying and understanding attack campaigns, and implementing preemptive measures. Traditional machine learning approaches face challenges such as labor-intensive campaign annotation, imbalanced attack data distribution, and concept drift. To address these challenges, we propose DYNAMO, a novel weakly supervised and human-in-the-loop machine learning framework for automated network attack attribution using raw network traffic records. DYNAMO integrates self-supervised learning and density-aware active learning techniques to reduce the overhead of exhaustive annotation, querying human analysts to label only a few selected highly representative network traffic samples. Our experiments on the CTU-13 dataset demonstrate that annotating less than 3% of the records achieves attribution accuracy comparable to fully supervised approaches with twice as many labeled records. Moreover, compared to classic active learning and semi-supervised techniques, DYNAMO achieves 20% higher attribution accuracy and nearly perfect detection accuracy for unknown botnet campaigns with minimal annotations.

1 INTRODUCTION

Cyber attack attribution aims to recognize the campaigns of attacks that are likely performed by the same organization and use similar attack techniques (Sahoo, 2022; Jaafar et al., 2020; Alrabaee et al., 2019; Zhang et al., 2019; Pitropakis et al., 2018; Nisioti et al., 2018; Rosenberg et al., 2017; Alrabaee et al., 2014). Deploying machine learning (ML) techniques for attack attribution harnesses the power of artificial intelligence (Lee and Choi, 2023; Ren et al., 2023; Haddadpajouh et al., 2020; Zhang et al., 2019; Rosenberg et al., 2017). The ML-based attribution methods excel at processing attack data with exponentially growing volumes and identifying automatically subtle patterns that may elude human analysts. By analyzing many data sources, ML-based attack attribution techniques can significantly enhance the accuracy and efficiency of attribution efforts. They can swiftly process and correlate indicators of compromise to identify commonalities across disparate attacks. Our study focuses on network attack attribution, identifying the campaigns responsible for malicious activities.

[a] https://orcid.org/0009-0001-4237-9587
[b] https://orcid.org/0000-0002-9035-6718

Traditional machine learning (ML) approaches encounter three primary challenges in network attack attribution. **First**, ML-driven methods necessitate substantial annotation efforts to construct a fully labeled training dataset for attributing attack campaigns (Rosenberg et al., 2017). The effectiveness of ML models, especially Deep Neural Networks, is linked to the availability of abundant labeled training samples. However, manual annotations and investigations become prohibitively costly.

Secondly, imbalances in data volumes across various attack campaigns introduce severe statistical bias for attack attribution (Sahoo, 2022; da Silva Freitas Junior and Pisani, 2022). The complexity of network attacks directly influences the volume of generated attack data, with specific techniques, such as brute-force attacks, denial-of-service attacks, and ransomware attacks, yielding extensive observation data. Conversely, more sophisticated attack campaigns may go unnoticed due to their limited generation of dispersed logs over an extended period.

Given the intrinsic imbalanced data distribution, the most frequently occurring campaigns are more likely to be gathered and annotated. Consequently, the attack attribution model trained with highly imbalanced data will easily overfit the majority campaigns

Orsini, H. and Han, Y.
DYNAMO: Towards Network Attack Campaign Attribution via Density-Aware Active Learning.
DOI: 10.5220/0012759100003767
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 21st International Conference on Security and Cryptography (SECRYPT 2024)*, pages 91-102
ISBN: 978-989-758-709-2; ISSN: 2184-7711
Proceedings Copyright © 2024 by SCITEPRESS – Science and Technology Publications, Lda.

while ignoring the minority ones. **Thirdly**, the ever-evolving nature of cyber-attacks poses a challenge in the form of concept drift (Yang et al., 2021). The tactics and infrastructure of attack campaigns continually evolve to evade detection and exploit new vulnerabilities. A ML-driven attack campaign attribution system trained on historical data may no longer be relevant if there is a behavior shift.

Echoing the challenges, our study proposes a novel framework of weakly supervised and human-in-the-loop ML-based attack attribution, known as *DYNAMO*, to address the bottlenecks. As in Figure 1, *DYNAMO* comprises three modules.

Similarity-Enhancing and Self-Supervised Feature Learning with Unlabeled Network Traffic Data. The initial module employs self-supervised learning (Wen and Li, 2021) to acquire a similarity-enhancing representation of unlabeled network traffic records. This process aims to encode raw network traffic data into a concise feature representation that groups network traffic data with similar profiles while separating those with distinct ones (Sahoo, 2022; Rosenberg et al., 2017; Lee and Choi, 2023). Modern network attacks are commonly automated by preprogrammed malware families and directed by commands from command-and-control servers. Consequently, network traffic data generated by the same campaign tends to exhibit similar patterns. Compared to raw network traffic, the similarity-enhancing latent features can better differentiate attack behaviors.

Density-Aware Active Learning to Select Representative Attack Data. Building on the learned feature representation, the pipeline incorporates a density-aware active learning module to sample and label a fraction of network traffic data. Initially, *DYNAMO* conducts clustering of network traffic data in the learned feature space. These derived clusters represent typical attack behaviors among the unlabeled data. Subsequently, *DYNAMO* samples a few representative network traffic records from each cluster and annotates them. This density-aware active learning process combines the methodologies of uncertainty sampling and a round-robin-based ranking method to prioritize clusters of smaller sizes during the sampling operation.

The primary objectives of this density-aware active learning module are threefold. Firstly, labeling network traffic records from each local cluster ensures comprehensive coverage of diverse attack behaviors in the unlabeled data pool. Secondly, the uncertainty sampling method in *DYNAMO* focuses on attack behaviors that remain underfit and uncertain to the attack attribution model, significantly enhancing the accuracy of attack attribution with minimal labeling overheads. Thirdly, the round-robin-based ranking method provides a balanced sample coverage across high and low-density areas in the unlabeled dataset.

Attack Attribution and Unseen Campaign Detection with Minimal Annotation Overheads. Compared to density-agnostic sampling strategies such as random selection or uncertainty sampling (Lewis and Catlett, 1994; Cohn et al., 1994), the density-aware active learning module constructs a more comprehensive and less biased labeled dataset for attack attribution. Leveraging the selected representative network traffic data, *DYNAMO* concurrently trains two models of attack attribution and unseen campaign detection tasks. With minimal labeling efforts, *DYNAMO* simultaneously achieves the identification of network traffic data belonging to human-annotated campaigns and the detection of the attack campaigns remaining unknown to human analysts in the pool of unlabeled network traffic records.

Difference Between Our Work and Intrusion Detection (IDS). The primary objective of DYNAMO is to identify distinct attack campaigns (Nisioti et al., 2018). DYNAMO takes a collection of malicious network traffic records as input. The output of DYNAMO determines whether the observed malicious traffic activities are associated with specific attack campaigns previously identified by human analysts or generated by a previously unseen campaign, distinct from the known ones. DYNAMO can be further chained with intrusion detection systems to detect and categorize attack behaviors. However, differentiating normal and malicious network activities is beyond the scope of this work.

We summarize our contribution as below:

First, we propose DYNAMO to achieve accurate attack attribution across campaigns with highly imbalanced distributions while minimizing campaign annotation overhead. Empirical results demonstrate that DYNAMO requires annotating 3% of the network traffic records to achieve a campaign attribution accuracy close to the fully supervised baseline trained with over 80% of the network traffic data. Compared to uncertainty sampling based on active learning (Lewis and Catlett, 1994; Balcan and Long, 2013; Zhou et al., 2003; Han and Shen, 2016), DYNAMO exhibits significantly higher campaign attribution accuracy. Especially, (Han and Shen, 2016) is adopted as an active learning solution to spear-phishing campaign attribution. DYNAMO shows over 10% higher attribution accuracy than this approach in the test.

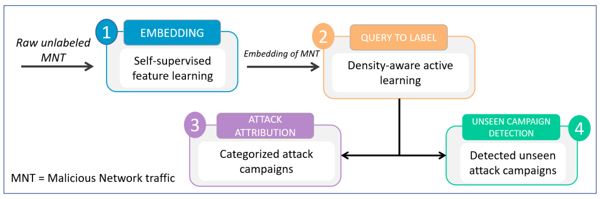
Second, with the active learning technique, DYNAMO accurately detects the campaigns unseen in the training phase of the campaign classifier while minimizing the labeling overheads over the samples from already recognized campaigns. Our experimental study shows that DYNAMO can achieve almost perfect detection accuracy of unknown botnet campaigns with less than 1% of the network traffic records annotated, which is 30% higher than two state-of-the-art anomaly detection-based baselines.

Third, we demonstrate DYNAMO's effectiveness using a large-scale botnet traffic dataset. This dataset contains over 444,699 botnet traffic flows from 13 botnet campaigns, with highly imbalanced data distribution across different campaigns. 4 of the 13 campaigns take over 90% of the network traffic records. We randomly divide the botnet campaign into two parts, i.e., we select 7 campaigns as known campaigns and the rest 6 as unseen ones to mimic concept drift of attack attribution. With this challenging setting, we measure DYNAMO's attack attribution and unseen campaign detection performance. The empirical results confirm the validity of the design of DYNAMO.

2 RELATED WORK

Accurate attack attribution plays a pivotal role in deterring future cyber threats by enabling the application of targeted defense mechanisms. In current security practices, attack attribution relies on synthesizing and analyzing threat intelligence reports (Pitropakis et al., 2018; Jaafar et al., 2020; Alrabaee et al., 2014). These reports are crafted by aggregating intelligence from diverse sources, including open-sourced intelligence, social media intelligence, human intelligence, and intelligence gathered from the deep and dark web (Sahoo, 2022; Pitropakis et al., 2018; Jaafar et al., 2020). Integrating these information sources helps unveil mechanisms, indicators, and actionable insights related to emerging cyber threats. However, the manual investigation-based attack attribution botteneck lies in the substantial domain-specific and hardware-dependent knowledge required from human security analysts to identify relevant indicators of attack behaviors across different campaigns. This results in significant costs of manual investigation of attacks, impeding timely responses to mitigate cyberattacks.

In contrast to manual investigation-based attack attribution, machine learning (ML)-driven methods offer automated categorization of attack campaigns based on security incident logs or network traffic patterns (Rosenberg et al., 2017; Lee and Choi, 2023; Nisioti et al., 2018; Zhang et al., 2019; Haddadpajouh et al., 2020). These ML-based solutions approach the attack attribution problem as a multi-class classification task, taking inputs like system logs (e.g., NetFlow records or sandbox analysis logs). These inputs are encoded into computable feature representations of attack behaviors. The classifiers are built over these features to categorize the encoded attack behaviors into pre-labeled attack campaigns, e.g., different malware authors and network attack campaigns. However, applying these ML-based attack campaign attribution methods requires to first preparing a fully supervised training set, i.e., a set of network traffic/system logs paired with explicitly identified campaigns. Annotating attack campaigns usually requires substantial manual investigation into the network traffic/system logs. Besides, The imbalanced and dynamically evolving nature of attack behaviors undermines the performance of ML-based attack attribution, violating the Independent and Identically Distributed (IID) assumption of ML methods (da Silva Freitas Junior and Pisani, 2022).



Raw unlabeled MNT
1
EMBEDDING
Self-supervised feature learning
Embedding of MNT
2
QUERY TO LABEL
Density-aware active learning
3
ATTACK ATTRIBUTION
Categorized attack campaigns
UNSEEN CAMPAIGN DETECTION
4
Detected unseen attack campaigns
MNT = Malicious Network traffic

Figure 1: DYNAMO-based workflow for attack attribution and unseen attack campaign detection.

Active Learning. Active learning aims to enhance classifier performance by strategically selecting unlabeled samples for labeling, typically focusing on uncertain or informative instances. These samples are then labeled by external sources and incorporated into the training set to refine the classifier. Methods in this domain often prioritize challenging regions near classifier boundaries in the feature space and employ uncertainty measures like posterior probability and prediction entropy (Lewis and Catlett, 1994; Cohn et al., 1994; Balcan and Long, 2013). Recent efforts extend active learning to spear-phishing campaign attribution (Han and Shen, 2016), refining the selection criteria for uncertain samples (Sinha et al., 2019; Deng et al., 2018). However, conventional active learning approaches struggle with imbalanced data distributions, particularly in neglecting minority class samples, which exacerbates misclassifications early in the learning process when uncertainty measures may be less reliable.

3 DESIGNING DYNAMO

Categorizing different attack campaigns hinges on assessing the similarity between network traffic patterns, particularly botnet attacks programmed by Command and Control servers. In these attacks, network traffic patterns within the same campaign tend to be very similar. In situations where limited network traffic is annotated with campaign labels, a high-quality similarity metric between network activities proves crucial for estimating the distribution of distinct campaigns. Furthermore, active learning is also pivotal in alleviating the challenge of limited annotated traffic. For imbalanced data distribution, active learning should prioritize sampling representative data from locally sparse areas in the data distribution, which are likely to contain traffic from rarely occurring campaigns. This density-aware sampling strategy balances the sample coverage over rarely and frequently occurring campaigns.

Following the principles, the DYNAMO's workflow, shown in Figure.1, operates for practical attack campaign attribution. Initially, it ingests a pool of unlabeled malicious network traffic records captured from System Information and Event Management (SIEM) system. During training, DYNAMO applies a nearest-neighbor-based self-supervised learning technique that compresses raw records into feature embeddings. Then, DYNAMO employs density-aware active learning on unlabeled records, strategically selecting representative samples. These samples are annotated by human analysts with campaign labels. DYNAMO concurrently builds an attack attribution and an unseen campaign detection model by leveraging these labels. In testing, DYNAMO reaches dual objectives: 1) categorizing network traffic into human-annotated attack campaigns; 2) detecting traffic critically distinct from human-annotated campaigns, potentially emanating from previously unseen campaigns. DYNAMO may then be reapplied to categorize the new campaigns.

3.1 Self-Supervised Feature Encoding

Let $X = \{x_i\}$ (i=1,2,3,...,N) denote the network traffic records of different attack campaigns. Each x_i can be raw network traffic data, such as NetFlow records, or aggregated statistics of NetFlow records within sliding time windows (Garcia et al., 2014; Kim et al., 2020). DYNAMO uses raw NetFlow records as the input. NetFlow aggregates pcap data into flow, providing an efficient and scalable metadata-based representation of communication. It includes numerical and categorical features such as source and destination IP addresses, ports, network protocols, traffic duration, etc (Sarhan et al., 2020). As reported in (Nisioti et al., 2018; Garcia et al., 2014), NetFlow is used widely in network traffic analysis and attack detection. Our study inherits the NetFlow features used for intrusion detection as in (Sarhan et al., 2020).

As shown in Figure.2, given the input network traffic data, this self-supervised learning module aims to map the raw NetFlow records into a low-dimensional latent feature space. The learned feature space is designed to enhance the similarity relation between network traffic data: pairs of network traffic records x_i and x_j sharing similar attributes are forced to stay close, thus having a higher similarity level. In contrast, pairs of network traffic with different patterns are separated as much as possible, holding a lower similarity level. To reach this goal we generate a K-Nearest Neighbor (KNN) graph over all the Netflows. In this KNN graph, each node is a NetFlow record x_i and linked to its K-Nearest Neighbors, i.e., $\mathrm{KNN}(x_i) = \{x_i^{NN,1}, x_i^{NN,2}, x_i^{NN,3}, ..., x_i^{NN,k}\}$. In our study, we empirically choose K to be 5, providing a sparse KNN graph structure and showing the optimal performance. We then train a GraphSage-based auto-encoder h parameterized by θ in DYNAMO to optimize the objective function in the following:

$$\begin{aligned}\theta^* = \arg\min_{\theta} - \frac{1}{nK}\sum_{i=1}^{n}[\sum_{k=1}^{K} log(\sigma(h_\theta^T(x_i)h_\theta(x_i^{NN,k}))) \\ -\lambda \sum_{j,x_v \notin \mathrm{KNN}(x_i)} log(\sigma(-h_\theta^T(x_i)h_\theta(x_v)))]\end{aligned} \quad (1)$$

where $x_v \notin \mathrm{KNN}(x_i)$ denote the nodes in the KNN graph that are not connected to x_i (hence beyond the 5-hop nearest neighbors of x_i). σ is the sigmoid function. In Eq.1, minimizing the first term maximizes the similarity between x_i and its nearest neighbors $\mathrm{KNN}(x_i)$ in the latent feature embedding space. Minimizing the second term maximizes the distance (minimizing similarity) between x_i and any data points beyond the K nearest neighbors. λ is a hyperparameter balancing the second term's impact in the objective function. Optimizing Eq.1 aims to generate a compact feature representation of raw NetFlow data that can separate different network flows as much as possible. In parallel, network flows with similar features tend to be produced by the same attack campaign. In the GraphSage-based feature space, these similar network flows are grouped together, which facilitates identifying the clusters of similar traffics. In conclusion, this GraphSage-based feature learning method trains a similarity-enhancing encoder of raw NetFlow. It swells the similarity between network traffic executed by the same attack campaign and difference between attack campaigns. We can then use the learned feature embeddings of NetFlow data to identify attack campaigns. We note that DYNAMO provides a flexible workflow: other self-supervised learning en-

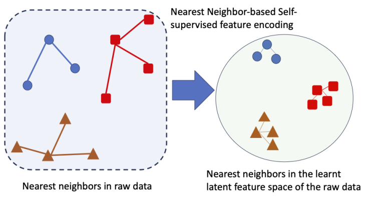

Nearest Neighbor-based Self-supervised feature encoding
Nearest neighbors in raw data
Nearest neighbors in the learnt latent feature space of the raw data

Figure 2: The concept of GraphSage-based self-supervised feature learning.

coders, such as triple contrastive learning, can be deployed in the feature encoding module of DYNAMO. We focus on demonstrating how the self-supervised learning-based encoder helps automated campaign attribution with weak supervision.

3.2 Density-Aware Active Learning

DYNAMO performs hierarchical clustering of the learned latent features of network traffic data $h_\theta(x_i)$ $(x_i \in X)$. Hierarchical clustering has been used frequently in data analysis (Silva et al., 2018), reaching good clustering results yet inducing reasonable overheads. The clustering algorithm aims to identify the groups of network traffic with highly similar profiles, which potentially belong to the same attack campaigns. More advanced clustering algorithms, e.g. spectral clustering and DBSCAN, can be integrated into DYNAMO. Empirically, we find these clustering methods end up with similar campaign attribution results. DYNAMO then conducts cluster-wise sampling. Network traffic selected from each cluster is annotated with campaign labels by external oracles, e.g., human analysts or MITRE attack knowledge graph. DYNAMO trains a classifier f parameterized by ψ to map an input NetFlow record to the annotated campaign label. Let $\mathbb{C} = \{C_1, ..., C_M\}$ denote the M clusters derived. The active learning process in DYNAMO iteratively executes two steps.
Initialization of the Attack Attribution. At the initial stage, DYNAMO selects data points $\Gamma_M = \{x_i^c\}$ (i=1,2,3,...,M) closest to the center of each cluster, queries the external oracle to obtain corresponding attack campaign labels y_i^c and initializes the classifier f using the initial labeled dataset $S = \Gamma_M$. **The round-robin sampling strategy.** In each iteration t of the active learning, DYNAMO aims to select a diverse set of network traffic records with the lowest decision confidence of the classifier f. These selected data points are added to the labeled training dataset S. By selecting samples that the classifier is least confident about, active learning aims to reduce uncertainty to the maximum extent in the classifier. These samples are underfitted by the classifier. They are often located in the regions where the decision boundary is ambiguous or

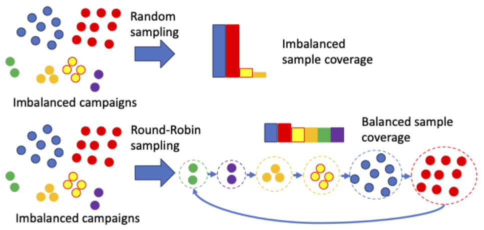


Figure 3: The Round-Robin sampling strategy for density-aware active learning.

where the model is likely to make errors. Including such instances in the training set can lead to a more robust and accurate model.

Figure.3 demonstrates the round-robin sampling process. We use the certainty score $u(x_i)$ of an input network traffic record x_i (Lewis and Catlett, 1994) to measure the decision confidence, i.e., a lower certainty score denotes lower decision confidence. DYNAMO hence applies the classifier f over the whole pool of unlabeled network traffic records $X_{\text{unlabeled}} = XS$ and selects Q records with the lowest certainty score. After that, DYNAMO retrieves the clusters that contain these Q records in the total M clusters. We note these clusters containing the Q data records with the highest uncertainty scores as $\hat{\mathbb{C}} = \{\hat{C}_1, \hat{C}_2, ..., \hat{C}_{M_q}\}$ with $M_q < M$. DYNAMO sorts the M_q clusters in an ascending order of the cluster size. The sorted clusters are given as $\hat{C}_{i_1}, \hat{C}_{i_2}, ..., \hat{C}_{i_{M_q}}$, where $i_1, i_2, ..., i_{M_q}$ are the indices of the clusters in the sorted cluster list. $|\hat{C}_{i_1}| \leq |\hat{C}_{i_2}| ... \leq |\hat{C}_{i_{M_q}}|$. DYNAMO employs a round-robin sampling method. It follows a cyclic sampling process, starting from the first (the smallest) cluster $\hat{C}_{i_1}$. For each of the M_q clusters, it selects one network traffic record with the lowest certainty score to annotate and moves to the next cluster along the ranked list of clusters. After obtaining a sample from the largest cluster $\hat{C}_{i_{M_q}}$, it returns to the smallest cluster with still unlabeled data records. This process repeats until the maximum number of the labeled network traffic records is reached for the iteration t. The selected records are added to S to update the classifier.

This round-robin sampling strategy uses the cluster's size to evaluate the cluster's distribution density in the latent feature space. A cluster of a smaller size represents a sparser area and is more likely to include rarely appearing campaigns compared to high-density areas. The round-robin method guarantees that DYNAMO prioritizes the sparse areas containing informative examples over the dense areas in the sampling process. It balances the number of training samples from minority and majority campaigns.

3.3 Detecting Unseen Campaigns

In parallel with training the classifier f for attack attribution, DYNAMO builds an unseen attack campaign detector g parameterized by ϕ in the latent feature space, using the labeled network traffic records S selected by the density-aware active learning module. The network traffic records annotated by human analysts are considered samples from the attack campaigns already known. In the pool of remaining unlabeled network traffic records, the task of unseen campaign detection is to decide whether an unlabeled network traffic record belongs to any labeled campaigns or a new campaign beyond the labeled samples. Formally, the detector maps the latent feature embedding of a network traffic record x to a binary response, i.e.,$g_{\phi}(h_{\theta}(x)) \rightarrow \{1,-1\}$. with 1 denoting unseen campaigns and -1 for known campaigns.

We incorporate two state-of-the-art one-class anomaly detection methods (Moya and Hush, 1996), One-class SVM and Isolation Forest, for the unseen campaign detection task. These methods have proven effective in detecting concept drift in intrusion detection, malware classification systems, and outlier detection (Nisioti et al., 2018; Karev et al., 2017; Burnaev and Smolyakov, 2016). In DYNAMO, we adhere to the one-class classification setting (Moya and Hush, 1996) and train both one-class anomaly detection models using annotated network traffic records from known campaigns. In this setup, the selected labeled network traffic records S serve as the representative dataset for known attack campaigns. The two one-class anomaly detection methods establish a hypersphere in the latent feature space to encompass data points in S. During testing, points inside the sphere are considered inliers (data from known campaigns), while points outside are flagged as outliers from potentially unknown campaigns. However, these one-class methods are sensitive to the high diversity of training data, often classifying rarely-appearing classes of normal data as anomalies and leading to false alarms. To address this issue, we propose using Positive-Unlabeled (PU) learning (Plessis et al., 2015) to enhance unseen campaign detection.

PU learning formulates unseen campaign detection as a binary classification task. The labeled network traffic data S and the remaining unlabeled records X/S are treated as positive and negative training data. As X/S is potentially a mixture of the network traffic data of the known and unseen attack campaigns, they form a set of noisy negative training data. Our work adopts the PU learning method in (Plessis et al., 2015) to recover the boundary differentiating the known campaigns from previously unseen campaigns. Integrating PU learning and active learning in DYNAMO for unseen campaign detection offers a two-fold advantage. First, PU learning explicitly includes labeled training samples of known campaigns, allowing DYNAMO to capture accurately the characteristics differentiating known and unseen campaigns. Second, PU learning benefits from density-aware active learning, using labeled representative network traffic records to provide balanced coverage over both majority and minority attack campaigns. This approach can better capture variability within known attack campaigns, reducing sensitivity to imbalanced campaign distribution and improving the performance of unseen campaign detection.

4 EXPERIMENTAL STUDY

We demonstrate the use of DYNAMO with the task of botnet campaign attribution. Our experimental study shows the merits of DYNAMO by addressing the following 3 questions:

Question 1. (Q1) Compared to using raw NetFlow data directly, can the self-supervised feature learning technique of DYNAMO improve the performance of attack attribution ?

Question 2. (Q2) Highly imbalanced network traffic data distribution from various attack campaigns may pose challenges to the density-agnostic query-to-learn active learning techniques. Compared to them, can DYNAMO's density-aware active learning module help reach more effective attack attribution?

Question 3. (Q3) Except for a few NetFlow records annotated with the corresponding campaigns, plenty NetFlow data remain unannotated due to the expensive cost of manual investigation. Can unannotated data be useful to boost the performance of unseen campaign detection ?

4.1 Experimental Setup

Dataset. We use the *CTU-13* dataset (Garcia et al., 2014) as the benchmark data to evaluate the effectiveness of DYNAMO. The CTU-13 is a dataset of botnet traffic records captured by CTU University in 2011. It contains 13 scenarios of botnet attacks. Each botnet scenario was defined as a particular infection of the virtual machines by executing a specific type of malware. Across different botnet scenarios, different network protocols were employed for botnets, and different attack actions such as IRC-based, PortScan, Spam, DDoS attacks were adopted. For example, botnet scenario 10 primarily employs IRC botnets and DDoS attacks, while botnet scenario 9 also introduces parallel PortScan, ClickFraud, and Spam-based attacks. The diversity of the attack actions results in different attack behaviors between botnet scenarios. In the following experiments, *we treat each scenario as a separate attack campaign*. The task of attack attribution in the following experiments is categorizing the NetFlow records to the corresponding scenarios. In total, there are $444,699$ NetFlow records of botnet flows in the whole 13 scenarios. We use the scenario label of each NetFlow record as the ground truth of attack attribution and unseen campaign detection. As shown in (Garcia et al., 2014), different botnet scenarios contribute drastically varied numbers of NetFlow data. The 1st, 3rd, 9th, and 10th botnet scenarios contain over 85% of the NetFlow data in the whole dataset. Since differentiating botnet traffic from benign ones is beyond our scope, we do not use either benign or background traffic in CTU-13.

The Test Settings. We follow the dataset split setting in (Garcia et al., 2014; Kim et al., 2020) in the attack attribution test. We pick 7 out of the total 13 botnet scenarios (the 3,4,5,10,11,12,13 scenarios). These 7 scenarios were captured by executing the botnet Rbot, Virus, and NSIS.ay. They perform IRC-based, P2P-based and HTTP-based communication methods including botnet attacks such as Spam, ClickFraud, PortScan, DDoS, and FastFlux. The rest 6 campaigns (the 1,2,6,7,8, and 9 scenarios) were executed with the botnet Neris, Sougou, Menti, and Murlo. The NetFlow records in the 7 and 6 campaigns are noted as $\mathcal{D}_{\text{attr}}$ and $\mathcal{D}_{\text{ood}}$. They contain 186990 and 257709 NetFlow records, respectively. The botnet malware samples used to generate the two subsets of scenarios have no overlapping. This split aims to mimic the real-world situation where the captured botnet attacks are potentially launched by different attack campaigns in term of protocols and attack behaviors. Using $\mathcal{D}_{\text{attr}}$ and $\mathcal{D}_{\text{ood}}$, we define two testing settings: *Attack Attribution* and *Unseen Campaign Detection*.

Attack Campaign Attribution. We randomly select 80% of NetFlow data in each of the 7 scenarios in $\mathcal{D}_{\text{attr}}$ as the training set, noted as $\mathcal{D}_{\text{attr}}^{\text{train}}$. The rest 20% NetFlow samples of each scenario in $\mathcal{D}_{\text{attr}}$ are used as the testing set, noted as $\mathcal{D}_{\text{attr}}^{\text{test}}$. The botnet scenario labels of $\mathcal{D}_{\text{attr}}^{\text{test}}$ are taken as the ground truth attack campaign labels for campaign attribution test.

DYNAMO selects $p\%$ of $\mathcal{D}_{\text{attr}}^{\text{train}}$ to label by human experts with $p = 0.7\%,1.3\%,2.0\%,2.6\%$, corresponding to 1000,2000,3000,4000 and 5000 samples respectively. DYNAMO trains the classifier with the selected and labeled NetFlow records. Then DYNAMO applies the classifier over $\mathcal{D}_{\text{attr}}^{\text{test}}$ to measure the accuracy of the attack attribution classification. The higher the botnet scenario classification accuracy is, the more effective DYNAMO is for attack attribution.

Unseen Campaign Detection: We consider the botnet scenarios in $\mathcal{D}_{\text{attr}}$ and $\mathcal{D}_{\text{ood}}$ as known and unseen campaigns. To demonstrate that the unseen campaign detection can be conducted in parallel to the attribution of known campaigns, we provide to DYNAMO $p\%$ of the NetFlow records in $\mathcal{D}_{\text{attr}}^{\text{train}}$ (the training data of attack attribution) annotated with the attack campaign labels. They are considered labeled samples from known attack campaigns. We note this selected subset as $\mathcal{D}_{\text{attr}}^{\text{train,labeled}}$. The rest unlabeled traffic data in $\mathcal{D}_{\text{attr}}^{\text{train}}$ are noted as $\mathcal{D}_{\text{attr}}^{\text{train,unlabeled}}$.

We randomly split $\mathcal{D}_{\text{ood}}$ into two non-overlapped subsets $\mathcal{D}_{\text{ood}}^{\text{train}}$ and $\mathcal{D}_{\text{ood}}^{\text{test}}$, containing 80% and 20% of NetFlow records in $\mathcal{D}_{\text{ood}}$. We train One-Class SVM-based and Isolation Forest-based detector as two baselines using the labeled samples $\mathcal{D}_{\text{attr}}^{\text{train,labeled}}$. For the PU learning module of DYNAMO, we train g_{ϕ}^{pu} using $\mathcal{D}_{\text{attr}}^{\text{train,labeled}}$ as positive training data. We combine $\mathcal{D}_{\text{attr}}^{\text{train,unlabeled}}$ and $\mathcal{D}_{\text{ood}}^{\text{train}}$ together to form the noisy negative training data of PU learning, noted as $\mathcal{D}^{\text{unlabeled}}$. In this setting, $\mathcal{D}^{\text{unlabeled}}$ contains a mixture of both already known and unknown botnet campaigns. We evaluate the performance of *unseen campaign detection* on $\mathcal{D}_{\text{attr}}^{\text{test}}$ and $\mathcal{D}_{\text{ood}}^{\text{test}}$. **Evaluation metric.** We utilize *Macro F1* to measure the attack campaign attribution accuracy and unseen campaign detection accuracy. To report the performance metric, we repeat the random split of $\mathcal{D}_{\text{attr}}$ into $\mathcal{D}_{\text{attr}}^{\text{train}}$ and $\mathcal{D}_{\text{attr}}^{\text{test}}$ for 10 times. The average and standard deviation of each metric is reported in the experimental results.

The Settings of DYNAMO. DYNAMO projects a raw NetFlow record to a 64 dimensional embedding vector using the self-supervised feature encoder, which empirically provide the optimal performance. In each round of active learning, DYNAMO selects at most 50 NetFlow records. We emphasize that proposing a new classifier architecture for attack attribution is beyond our scope. To demonstrate the application of DYNAMO, we choose Gradient Boosted Trees (**GBT**) composing 800 trees for attack attribution and unseen campaign detection. GBT-based classifiers have been widely used in various cyber security applications. It provides competitive and accurate classification performances compared to more complex models, e.g., deep neural networks. We also involve Label Spreading (**LS**) as a baseline in our study, which is a semi-supervised classifier previously used for spear-phishing campaign attribution. It has been combined with uncertainty sampling-based active learning for spear-phishing campaign attribution in (Han and Shen, 2016). It propagates class label confidence scores across nearest neighbors to estimate the class label of unlabeled data points. Compared to GBT, LS is sensitive to the imbalanced class distribution. For unseen campaign detection, we inherit the same hyperparameter settings of GBT in the attack attribution test. We use 800 trees for Isolation Forest and the RBF kernel for One-class SVM.

4.2 Attack Campaign Attribution

The empirical study provides the answer to the question **Q1** and **Q2**. We involve three alternative baselines to DYNAMO.

Attack Attribution with Full Supervision. We use a fully supervised GBT classifier, trained on the entirety of the attack-labeled training dataset $\mathcal{D}_{\text{attr}}^{\text{train}}$. Proximity of campaign classification accuracy between DYNAMO and this baseline indicates that DYNAMO can provide accurate campaign attribution.

UAL. The uncertainty sampling-based active learning method (UAL) (Lewis and Catlett, 1994; Han and Shen, 2016) performs an iterative annotation-retraining process. In each iteration, it first selects and annotates the 50 network traffic records with the highest uncertainty scores in the unlabeled data of $\mathcal{D}_{\text{attr}}^{\text{train}}$. These annotated records are added to the training dataset. The classifier is then retrained using the enriched and fully annotated training dataset.

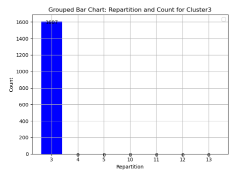
Random Selection. We randomly select network traffic data from the whole pool of $\mathcal{D}_{\text{attr}}^{\text{train}}$ to label and train the classifier.

We implement DYNAMO and all the baselines using both raw data and the learned feature space. We compare the attack attribution performance with/without the learned features to confirm the merits of introducing the self-supervised feature learning module. Table.1 and Table.2 present the averaged and standard deviation of Macro F1 score of attack attribution of all the involved methods using the raw data and the learned features, respectively. In each table, we vary the number of the selected samples from the training dataset $\mathcal{D}_{\text{attr}}^{\text{train}}$ in the active learning process. In Table.1 and Table.2 , we consistently observe superior attack attribution accuracy achieved using the DYNAMO's learned features. Across various fractions of labeled data, Macro F1 scores of Random Selection, DYNAMO, and UAL using learned features are, on average, 15% higher than those obtained using raw NetFlow data. Notably, using learned features, both DYNAMO and UAL can achieve Macro F1 scores close to the full supervision method employing the entire training dataset $\mathcal{D}_{\text{attr}}^{\text{train}}$, requiring only 3% of the training dataset (5000 labeled samples). These empirical observations confirm the accuracy-boosting effect of self-supervised learning, which answer **Q1** raised before.

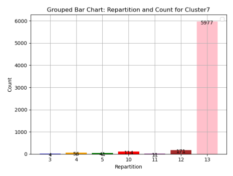
DYNAMO performs clustering of NetFlow records in the learned feature space. To intuitively illustrate the merit of the self-supervised learning module, we show 6 out of the whole 50 clusters in Figure.4. In each plot, the x-axis represents the botnet campaign labels. The y-axis gives the number of the NetFlow records attributed to different campaigns in each cluster. As shown, the NetFlow data in all the 6 clusters are dominated by only one botnet campaign. It indicates that each cluster contains highly similar network traffic patterns. In the learned feature space, similar network traffic flows are compressed into close feature embeddings and different flows are separated with a distinctive gap. This similarity-enhancing characteristic of the learned feature space facilitates the identification of representative network traffic patterns, prompting the performance of attack attribution. In Cluster 13, except for 353 NetFlow records from botnet scenario 4, there are also 84 NetFlow records from botnet scenario 13. One potential reason for the overlapping can be that both botnet scenarios involve traffic for communication with C2 servers and data exfiltration, which show similar network traffic patterns. Cluster 34 contains 77 and 320 NetFlow records from botnet scenarios 5 and 13. Both scenarios are executed by Virut malware for spam and port scan attacks, leading to similar traffic patterns.

We address **Q2** by examining the outcomes from various perspectives, as outlined in Table 1 and Table 2 for Macro-F1. As shown in Table 2, the average Macro-F1 scores of DYNAMO surpass those of Random Selection by up to 15% when utilizing the learned feature space across varying fractions of labeled data points. Simultaneously, Figure.5 illustrates the average percentages of labeled NetFlow records from different campaigns using DYNAMO and Random Selection. Notably, DYNAMO yields significantly more balanced labeled NetFlow records covering diverse botnet campaigns than those selected by Random Selection. These observations mutually reinforce one another: DYNAMO supplies less biased labeled samples for training the attack attribution model, achieving superior accuracy compared to Random Selection.

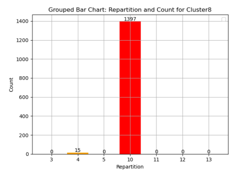
As depicted in Table.1, DYNAMO consistently achieves a higher average Macro-F1 with 1/20th of the standard deviation over the Macro-F1 scores on raw NetFlow data compared to UAL. For Table.2, DYNAMO exhibits higher average Macro-F1 scores than UAL, especially when the number of the labeled NetFlow records is limited, e.g., less than 4000. For example, with 3000 records labeled, DYNAMO's Macro-F1 score is already close to the full supervi-



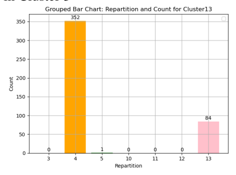
(a) Distributions of different campaigns in Cluster 3



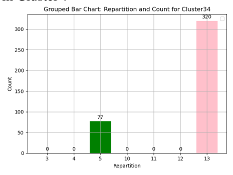
(b) Distributions of different campaigns in Cluster 7



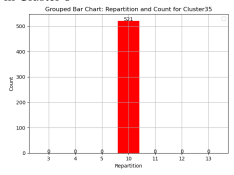
(c) Distributions of different campaigns in Cluster 8



(d) Distributions of different campaigns in Cluster 13



(e) Distributions of different campaigns in Cluster 34



(f) Distributions of different campaigns in Cluster 35

Figure 4: Distributions of different campaigns in 6 clusters derived by DYNAMO.

sion method using 80% of the training data. It is 6% and 14% higher than the averaged Macro-F1 score of UAL and Random Selection, respectively. Figure.5 displays the average percentages of network traffic data from different campaigns (y-axis) in the labeled NetFlow records using DYNAMO, UAL, and Random Selection with 1000, 2000, 3000, 4000, and 5000 labeled NetFlow records (x-axis). For Random Selection, the number of labeled samples from the minority botnet campaigns is generally less than 10, with some scenarios having only 1 labeled sample even when the total number of labeled data reaches 5000. In contrast, DYNAMO provides more than 20 labeled samples for minority campaigns. The balanced sampling coverage of DYNAMO results in more accurate attack attribution than Random Selection. Compared to UAL, DYNAMO reaches a more stable and balanced data distribution between the minority campaigns (e.g., botnet scenarios 3,4,5,11, and 12) and the dominant ones (botnet scenarios 10 and 13). In addition, the campaign distribution in the selected NetFlow records by UAL contains more drastic fluctuation than DYNAMO. This results in UAL's attribution accuracy being less accurate. The result affirms the merits of DYNAMO, particularly in scenarios with a tight labeling budget.

Furthermore, employing the GBT-based classifier, DYNAMO yields 10% to 15% higher average Macro-F1 scores than those obtained with the LS-based semi-supervised learning method used in (Han and Shen, 2016), irrespective of the sampling strategy and the number of labeled samples. The LS method iteratively propagates class membership confidence from labeled training samples to their unlabeled k-hop nearest neighbors. It is intrinsically sensitive to imbalanced class distribution, as the majority campaigns influence the estimated label confidence more. In contrast, GBT is composed of ensemble tree-based classifiers, which exhibit more resilience to data imbalance than the LS-based method. This result suggests the efficacy of combining the density-aware sampling strategy with class imbalance-resilient classifiers for attack attribution.

The Best v.s. Worst Classified Botnet Campaigns. We compute the class-wise F1 scores of DYNAMO with 5000 labeled NetFlow records. The two best-classified campaigns are botnet scenarios 3 and 13, with class-wise F1 scores of 0.999 and 0.990. NetFlow records in these two scenarios are almost perfectly classified to the correct campaigns. Botnet scenario 3 contains IRC botnet attacks executed by Rbot IRC bots. The network activity includes IRC C2 server communication and port scans. They can be characterized by the use of IRC ports that are rarely used in other scenarios (e.g., TCP port 6667). Besides, some of the traffic in this scenario is executed by the authors of CTU-13 to simulate attacks. These behaviors make this scenario easily differentiated from the others using ICMP/UDP/HTTP for spam and DDoS attacks. Botnet scenario 13 involves spam attacks executed by Virut with the attempts bypassing CAPTCHA on webmail servers. These behaviors are different from the DDoS/IRC botnet attacks. The worst two classified campaigns are botnet scenarios 5 and 11. Botnet scenario 5 is mostly misclassified to Botnet scenario 13. Though scenarios 5 and 13 are dedicated to different attack behaviors, as indicated by (Garcia et al., 2014), they are both executed by Virut for spam attacks. These two campaigns share similar network activities, e.g., they have similar C2 server communication traffic. Similarly, botnet scenarios 11 and 10 are both executed by Rbot for DDoS attacks. They also share similar network traffic patterns pre-programmed by Robt. Differentiating these two campaigns thus becomes difficult. Distinguishing scenarios 5 and 11 requires further investigating and encoding the payloads of communications, e.g., extracting and encoding C2 command strings in the payloads. This is beyond the scope of our current study, but definitely within our future plan.

4.3 Unseen Campaign Detection

We involve the following tests to provide the answer to the question **Q3**. For DYNAMO, UAL, and Random Selection, we implement Isolation Forest (ISO) and One-class SVM (OCSVM) as alternatives to the PU-learning-based unseen campaign detector. All of the 9 settings are evaluated over the testing NetFlow data from the 7 campaigns in $\mathcal{D}^{\text{test}}_{\text{attr}}$ (the known campaigns) and the other 6 campaigns in $\mathcal{D}^{\text{test}}_{\text{ood}}$ (the unseen campaigns). Table.3 and Table.4 report the average and standard deviation of Macro-F1 achieved in the 9 settings using raw NetFlow data and the latent feature space encoded within DYNAMO. The detection performance for unseen campaigns, detailed in Tables 3 and 4, reveals a substantial improvement when utilizing progressively more labeled NetFlow data from the seven known campaigns during the training phase. In comparison to Isolation Forest and One-class SVM models, it is consistently evident that DYNAMO's PU-learning-based detector achieves significantly higher detection accuracy. When employing raw NetFlow data, DYNAMO, with the PU-learning technique, exhibits orders of magnitude improvement and a 10% to 50% increase in Macro F1 compared to the Isolation Forest and OCSVM-based detection methods. By further leveraging feature embeddings, DYNAMO with the PU-learning technique achieves perfect detection accuracy (Macro-F1 of 1.0) with only 0.3% of the NetFlow records (1000 NetFlow records) labeled from the known campaigns. The Macro-F1 scores presented in Table 1 demonstrate a substantial deterioration in performance when applying the PU-learning technique to randomly selected labeled data with raw NetFlow records, reaching almost half of the scores achieved by ISO and OCSVM. In contrast, leveraging the PU-learning method with the active learning module of DYNAMO and UAL yields significantly higher detection accuracy. This highlights that relying solely on the PU-learning technique does not ensure precise unseen campaign detection. The effectiveness of PU learning is contingent on the representativeness of the labeled NetFlow records. Therefore, it becomes imperative to integrate the self-supervised feature encoder, the density-aware active learning module, and the PU-learning technique to ensure optimal detection performance. In the unseen campaign detection task, known campaigns display diverse network traffic patterns, but the scarcity of labeled data hampers consistent and accurate detection by both methods. In contrast, the PU learning module of DYNAMO leverages both labeled and unlabeled network traffic data, which provides an unbiased and direct estimate of the classification boundary between known and unseen campaigns, thereby enhancing DYNAMO's detection accuracy.

Tables 3 and 4 illustrate that leveraging features learned by the self-supervised module significantly enhances Macro-F1 scores compared to using raw NetFlow data. Moreover, employing these learned features elevates the AUC scores of all methods close to 1, indicating high performance. These findings consistently demonstrate substantial improvements in detection accuracy with the self-supervised module. Compared to density-agnostic strategies (UAL and Random Selection), utilizing density-aware techniques based on learned latent feature embeddings achieves the highest detection accuracy across all three detection models (ISO, OCSVM, and PU). This highlights the effectiveness of integrating the three key modules in DYNAMO, not only for categorizing network activities from various campaigns but also for identifying emerging campaigns.

5 CONCLUSION

In conclusion, we present DYNAMO, a weakly supervised machine learning pipeline for automated network attack attribution, circumventing the need for exhaustive campaign labeling. DYNAMO effectively addresses the three-fold challenge in ML-based campaign attribution, i.e. the limited labeled campaigns for training, imbalanced campaign distributions, and the emergence of unseen attack campaigns. Empirical results demonstrate DYNAMO's capability to accurately attribute attack data to known campaigns while concurrently detecting previously unknown campaigns. Future endeavors will focus on extending DYNAMO's applicability to categorize APT attack campaigns and explore self-supervised techniques for campaign identification to further enhance the autonomy of attack campaign attribution.

ACKNOWLEDGEMENT

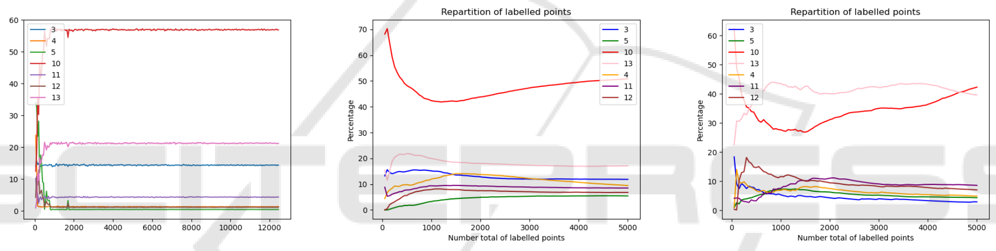
This work is funded by ANR PEPR project Superviz (22-PECY-0008) and ANR PEPR project DefMal (22-PECY-0007). We would also appreciate insightful discussion and suggestions offered by Professor Valerie Viet Triem Tong of CentraleSupelec.

Table 1: Mean and standard deviation (Mean ∓ standard deviation) of Macro F1 for attack attribution with *raw NetFlow data. The full supervision method trained using all training data achieves an average Macro F1 of* 0.674. NB: the number of the selected network traffic data.

	Attack attribution with the raw network traffic data					
	Random Selection		DYNAMO		UAL	
NB	GBT	LS	GBT	LS	GBT	LS
1000 (p=0.7%)	0.649 ∓ 0.022	0.576 ∓ 0.034	**0.654 ∓ 0.001**	0.576 ∓ 0.000	0.578 ∓ 0.043	0.533 ∓ 0.052
2000 (p=1.3%)	0.661 ∓ 0.010	0.573 ∓ 0.019	**0.654 ∓ 0.002**	0.580 ∓ 0.000	0.609 ∓ 0.055	0.573 ∓ 0.071
3000 (p=2.0%)	0.664 ∓ 0.009	0.583 ∓ 0.017	**0.654 ∓ 0.002**	0.590 ∓ 0.000	0.618 ∓ 0.036	0.603 ∓ 0.076
4000 (p=2.6%)	0.666 ∓ 0.008	0.659 ∓ 0.015	**0.654 ∓ 0.002**	0.583 ∓ 0.000	0.635 ∓ 0.024	0.648 ∓ 0.051
5000 (p=3.3%)	0.666 ∓ 0.006	0.658 ∓ 0.020	**0.653 ∓ 0.002**	0.583 ∓ 0.000	0.640 ∓ 0.022	0.652 ∓ 0.052

Table 2: Mean and standard deviation (Mean ∓ standard deviation) of Macro F1 for attack attribution with *the learned embedding features. The full supervision method trained using all of the data achieves an average Macro F1 of* 0.805. NB: the number of the selected network traffic data.

	Attack attribution with the latent feature learned by the self-supervised learning module					
	Random Selection		DYNAMO		UAL	
NB	GB	LS	GB	LS	GB	LS
1000 (p=0.7%)	0.611 ∓ 0.024	0.637 ∓ 0.036	**0.695 ∓ 0.024**	0.631 ∓ 0.000	0.607 ∓ 0.016	0.574 ∓ 0.067
2000 (p=1.3%)	0.653 ∓ 0.018	0.694 ∓ 0.022	**0.745 ∓ 0.021**	0.677 ∓ 0.000	0.613 ∓ 0.016	0.608 ∓ 0.017
3000 (p=2.0%)	0.673 ∓ 0.017	0.712 ∓ 0.016	**0.764 ∓ 0.016**	0.688 ∓ 0.000	0.723 ∓ 0.013	0.654 ∓ 0.027
4000 (p=2.6%)	0.686 ∓ 0.013	0.723 ∓ 0.049	**0.781 ∓ 0.015**	0.707 ∓ 0.000	0.773 ∓ 0.002	0.689 ∓ 0.019
5000 (p=3.3%)	0.697 ∓ 0.013	0.732 ∓ 0.012	**0.791 ∓ 0.011**	0.708 ∓ 0.000	0.785 ∓ 0.009	0.702 ∓ 0.020



(a) Random Selection (b) DYNAMO (c) UAL

Figure 5: The percentage number of the labeled NetFlow records belonging to different botnet scenarios.

Table 3: Mean and standard deviation (Mean ∓ standard deviation) of Macro F1 of unseen campaign detection on *raw NetFlow data*. NB: the number of the selected network traffic data.

	Unseen campaign detection using raw NetFlow data								
	Random Selection			DYNAMO			UAL		
NB	ISO	OCSVM	PU	ISO	OCSVM	PU	ISO	OCSVM	PU
1000 (p=0.7%)	0.816 ∓ 0.017	0.764 ∓ 0.000	0.296 ∓ 0.000	0.758 ∓ 0.071	0.764 ∓ 0.026	**0.892 ∓ 0.001**	0.489 ∓ 0.160	0.567 ∓ 0.191	0.890 ∓ 0.002
2000 (p=1.3%)	0.808 ∓ 0.025	0.764 ∓ 0.000	0.296 ∓ 0.000	0.762 ∓ 0.039	0.731 ∓ 0.026	**0.893 ∓ 0.001**	0.494 ∓ 0.169	0.478 ∓ 0.162	0.891 ∓ 0.002
3000 (p=2.0%)	0.764 ∓ 0.009	0.762 ∓ 0.000	0.296 ∓ 0.000	0.811 ∓ 0.024	0.620 ∓ 0.169	**0.893 ∓ 0.002**	0.419 ∓ 0.114	0.370 ∓ 0.012	0.892 ∓ 0.002
4000 (p=2.6%)	0.801 ∓ 0.009	0.762 ∓ 0.000	0.296 ∓ 0.000	0.762 ∓ 0.007	0.585 ∓ 0.180	**0.893 ∓ 0.001**	0.459 ∓ 0.015	0.407 ∓ 0.118	0.892 ∓ 0.001
5000 (p=3.3%)	0.797 ∓ 0.030	0.764 ∓ 0.000	0.296 ∓ 0.001	0.749 ∓ 0.009	0.673 ∓ 0.157	**0.893 ∓ 0.001**	0.461 ∓ 0.015	0.482 ∓ 0.182	0.892 ∓ 0.001

Table 4: Mean and standard deviation (Mean ∓ standard deviation) of Macro F1 of unseen campaign detection on *the learned embedding features*. NB: the number of the selected network traffic data.

	Unseen campaign detection with the latent feature learned by the self-supervised learning module								
	Random Selection			DYNAMO			UAL		
NB	ISO	OCSVM	PU	ISO	OCSVM	PU	ISO	OCSVM	PU
1000 (p=0.7%)	0.748 ∓ 0.005	0.853 ∓ 0.000	1.000 ∓ 0.000	0.913 ∓ 0.007	0.921 ∓ 0.005	**1.000 ∓ 0.000**	0.832 ∓ 0.043	0.898 ∓ 0.013	1.000 ∓ 0.000
2000 (p=1.3%)	0.754 ∓ 0.005	0.762 ∓ 0.000	1.000 ∓ 0.000	0.905 ∓ 0.010	0.913 ∓ 0.006	**1.000 ∓ 0.000**	0.817 ∓ 0.044	0.880 ∓ 0.016	1.000 ∓ 0.000
3000 (p=2.0%)	0.672 ∓ 0.005	0.696 ∓ 0.000	1.000 ∓ 0.000	0.765 ∓ 0.009	0.909 ∓ 0.008	**1.000 ∓ 0.000**	0.789 ∓ 0.018	0.860 ∓ 0.021	1.000 ∓ 0.000
4000 (p=2.6%)	0.758 ∓ 0.007	0.687 ∓ 0.000	1.000 ∓ 0.000	0.897 ∓ 0.010	0.904 ∓ 0.010	**1.000 ∓ 0.000**	0.789 ∓ 0.046	0.848 ∓ 0.048	1.000 ∓ 0.000
5000 (p=3.3%)	0.754 ∓ 0.007	0.689 ∓ 0.026	1.000 ∓ 0.000	0.891 ∓ 0.009	0.898 ∓ 0.008	**1.000 ∓ 0.000**	0.794 ∓ 0.059	0.842 ∓ 0.068	1.000 ∓ 0.000

REFERENCES

Alrabaee, S., Debbabi, M., and Wang, L. (2019). On the feasibility of binary authorship characterization. *Digital Investigation*, 28:S3–S11.

Alrabaee, S., Saleem, N., Preda, S., Wang, L., and Debbabi, M. (2014). Oba2: An onion approach to binary code authorship attribution. *Digital Investigation*, 11:S94–S103. Annual DFRWS Europe.

Balcan, M.-F. and Long, P. (2013). Active and passive learning of linear separators under log-concave distributions. In *COLT*, volume 30 of *Proceedings of Machine Learning Research*, pages 288–316, Princeton, NJ, USA. PMLR.

Burnaev, E. and Smolyakov, D. (2016). One-class svm with privileged information and its application to malware detection. In *ICDMW*, pages 273–280, Los Alamitos, CA, USA. IEEE Computer Society.

Cohn, D., Ghahramani, Z., and Jordan, M. (1994). Active learning with statistical models. In *NIPS*, volume 7. MIT Press.

da Silva Freitas Junior, J. and Pisani, P. H. (2022). Performance and model complexity on imbalanced datasets using resampling and cost-sensitive algorithms. In *IWLID 2022*, volume 183 of *Proceedings of Machine Learning Research*, pages 83–97. PMLR.

Deng, Y., Chen, K., Shen, Y., and Jin, H. (2018). Adversarial active learning for sequences labeling and generation. In *IJCAI*, pages 4012–4018. International Joint Conferences on Artificial Intelligence Organization.

Garcia, S., Grill, M., Stiborek, J., and Zunino, A. (2014). An empirical comparison of botnet detection methods. *Comput. Secur.*, 45:100–123.

Haddadpajouh, H., Azmoodeh, A., Dehghantanha, A., and Parizi, R. M. (2020). Mvfcc: A multi-view fuzzy consensus clustering model for malware threat attribution. *IEEE Access*, 8:139188–139198.

Han, Y. and Shen, Y. (2016). Accurate spear phishing campaign attribution and early detection. In *ACM SAC 2016*, SAC '16, page 2079–2086, New York, NY, USA. Association for Computing Machinery.

Jaafar, F., Avellaneda, F., and Alikacem, E.-H. (2020). Demystifying the cyber attribution: An exploratory study. In *(DASC 2020*, pages 35–40.

Karev, D., McCubbin, C., and Vaulin, R. (2017). Cyber threat hunting through the use of an isolation forest. In *ICCST*, CompSysTech '17, page 163–170, New York, NY, USA. Association for Computing Machinery.

Kim, J., Sim, A., Kim, J., Wu, K., and Hahm, J. (2020). Transfer learning approach for botnet detection based on recurrent variational autoencoder. In *IWSNTAA*, SNTA '20, page 41–47, New York, NY, USA. Association for Computing Machinery.

Lee, I. and Choi, C. (2023). Camp2vec: Embedding cyber campaign with attck framework for attack group analysis. *ICT Express*, 9(6):1065–1070.

Lewis, D. D. and Catlett, J. (1994). Heterogeneous uncertainty sampling for supervised learning. In *ICML*, pages 148–156. Morgan Kaufmann.

Moya, M. M. and Hush, D. R. (1996). Network constraints and multi-objective optimization for one-class classification. *Neural Networks*, 9(3):463–474.

Nisioti, A., Mylonas, A., Yoo, P. D., and Katos, V. (2018). From intrusion detection to attacker attribution: A comprehensive survey of unsupervised methods. *IEEE Communications Surveys and Tutorials*, 20(4):3369–3388.

Pitropakis, N., Panaousis, E., Giannakoulias, A., Kalpakis, G., Rodriguez, R. D., and Sarigiannidis, P. (2018). An enhanced cyber attack attribution framework. In Furnell, S., Mouratidis, H., and Pernul, G., editors, *Trust, Privacy and Security in Digital Business*, pages 213–228, Cham. Springer International Publishing.

Plessis, M. D., Niu, G., and Sugiyama, M. (2015). Convex formulation for learning from positive and unlabeled data. In *ICML*, volume 37 of *Proceedings of Machine Learning Research*, pages 1386–1394.

Ren, Y., Xiao, Y., Zhou, Y., Zhang, Z., and Tian, Z. (2023). Cskg4apt: A cybersecurity knowledge graph for advanced persistent threat organization attribution. *IEEE TKDE*, 35(06):5695–5709.

Rosenberg, I., Sicard, G., and David, E. O. (2017). Deepapt: Nation-state apt attribution using end-to-end deep neural networks. In Lintas, A., Rovetta, S., Verschure, P. F., and Villa, A. E., editors, *Artificial Neural Networks and Machine Learning – ICANN 2017*, pages 91–99, Cham. Springer International Publishing.

Sahoo, D. (2022). *Cyber Threat Attribution with Multi-View Heuristic Analysis*, pages 53–73. Springer International Publishing, Cham.

Sarhan, M., Layeghy, S., Moustafa, N., and Portmann, M. (2020). Netflow datasets for machine learning-based network intrusion detection systems. *CoRR*, abs/2011.09144.

Silva, D., Dell'Amico, M., Hart, M., Roundy, K. A., and Kats, D. (2018). Hierarchical incident clustering for security operation centers. In *IDEA'18, August 20, 2018, London, England*.

Sinha, S., Ebrahimi, S., and Darrell, T. (2019). Variational adversarial active learning. In *ICCV*, pages 5971–5980, Los Alamitos, CA, USA. IEEE Computer Society.

Wen, Z. and Li, Y. (2021). Toward understanding the feature learning process of self-supervised contrastive learning. In *ICML*, volume 139 of *Proceedings of Machine Learning Research*, pages 11112–11122. PMLR.

Yang, L., Guo, W., Hao, Q., Ciptadi, A., Ahmadzadeh, A., Xing, X., and Wang, G. (2021). CADE: Detecting and explaining concept drift samples for security applications. In *USENIX Security 21*, pages 2327–2344. USENIX Association.

Zhang, L., Thing, V. L., and Cheng, Y. (2019). A scalable and extensible framework for android malware detection and family attribution. *Computers and Security*, 80:120–133.

Zhou, D., Bousquet, O., Lal, T., Weston, J., and Schölkopf, B. (2003). Learning with local and global consistency. In *NIPS*, volume 16. MIT Press.