Uncovering Lateral Movement Using Authentication Logs

Haibo Bian, Tim Bai, Mohammad A. Salahuddin, *Member, IEEE*, Noura Limam, *Member, IEEE*, Abbas Abou Daya, and Raouf Boutaba, *Fellow, IEEE*

***Abstract*—Network infiltrations due to advanced persistent threats (APTs) have significantly grown in recent years. Their primary objective is to gain unauthorized access to network assets, compromise system and data. APTs are stealthy and remain dormant for an extended period of time, which makes their detection challenging. In this article, we leverage machine learning (ML) to detect hosts in a network that are a target of an APT attack. We evaluate a number of ML classifiers to detect susceptible hosts in the Los Alamos National Lab dataset. We (i) scrutinize graph-based features extracted from host authentication logs, (ii) use feature engineering to reduce dimensionality, (iii) explore balancing the training dataset using over- and under-sampling techniques, (iv) evaluate numerous supervised ML techniques and their ensemble, (v) compare our classification model to the state-of-the-art approaches that leverage the same dataset, and show that our model outperforms them with respect to prediction performance and overhead, and (vi) perturb the attack patterns to study the influence of change in attack frequency and scale on classification performance, and propose a solution for such adversarial behavior.**

***Index Terms*—Machine learning, advanced persistent threat, intrusion detection, adversarial learning.**

I. Introduction

CYBER attacks have recently grown in sophistication, resulting in considerable damage to businesses. They not only result in financial losses, but also impact customer trust and churn. Typically, an attack initiates by compromising several hosts or user accounts within a network, and leaves backdoors to gain persistent access to internal assets. This type of attack is commonly known as an advanced persistent threat (APT). According to Kaspersky Lab, an APT campaign in 2019 affected over one million users who installed the ASUS Live Update utility [1]. Similarly, a cryptocurrency exchange firm, DragonEx, announced in 2019 that it has suffered USD 7.09 million in losses due to an APT attack [2]. Therefore, it is imperative to defend against APT-assisted network intrusions.

Lateral movement (LM) is a crucial phase in an APT attack, which follows after an intruder has gained persistent access to certain network resources (e.g., servers, end-hosts). The goal of LM is to infiltrate other resources and gain higher privileges inside the target network. This is typically achieved by stealing credentials or exploiting vulnerabilities of already compromised hosts. Interestingly, 50%–90% of employees have access to data that they no longer need [3], which is primarily due to poor security practises, such as the violation of the least privilege principle [4]. This increases the likelihood of an attacker penetrating the crucial network assets via LM. Therefore, it is vital to detect LM at an early stage.

An alternative to the traditional detection of successful intrusions, is to pro-actively identify covert signs of LM. This can potentially generate alarms even before a successful intrusion has occurred, leading to LM detection during early exploration. After acquiring footprints of such behaviour, administrators can get insights into the attack strategy. They can also identify system vulnerabilities that can help alleviate future attacks. However, unlike hosts that act as proxies during the attack, newly compromised or vulnerable hosts are fairly dormant and leave minimal footprint, e.g., events in authentication logs. Furthermore, in large enterprises with thousands of hosts, it is unlikely that an infiltration will compromise the majority of hosts. Typically, the number of compromised hosts will be minuscule in comparison to the network size, resulting in *sparse* malicious activities. These issues make early detection of LM challenging.

Machine learning (ML) [5], [6]-based methods are widely used for intrusion detection. An important step prior to training a ML model is feature extraction. These features act as discriminators for training and inference, and impact the performance of ML models. The most commonly employed features (i.e., for ML model training and inference) in ML-based intrusion detection are either network flow-based (e.g., number of packets, direction, packet size and inter-arrival statistics) or host event-based (e.g., authentication type, authentication frequency, and user names used during authentication). However, these features do not completely capture the host communication patterns that may expose additional aspects of malicious behavior. Graph-based features, derived from flow-level or host-level information to reflect

Manuscript received May 2, 2020; revised October 9, 2020 and December 24, 2020; accepted January 16, 2021. Date of publication January 25, 2021; date of current version March 11, 2021. This work was supported in part by the Royal Bank of Canada and in part by the Natural Sciences and Engineering Research Council of Canada Collaborative Research and Development Grant 530335. The associate editor coordinating the review of this article and approving it for publication was G. Casale. *(Corresponding author: Noura Limam.)*

Haibo Bian, Tim Bai, and Raouf Boutaba are with the Department of Computer Science, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: haibo.bian@uwaterloo.ca; tim.bai@uwaterloo.ca; rboutaba@uwaterloo.ca).

Mohammad A. Salahuddin is with the David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: mohammad.salahuddin@uwaterloo.ca).

Noura Limam and Abbas Abou Daya are with the D.C. School of Computer Science, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: n2limam@uwaterloo.ca; aaboudaya@uwaterloo.ca).

Digital Object Identifier 10.1109/TNSM.2021.3054356

the true behaviour of network hosts, are an alternative that can overcome this limitation. In [7], we show that incorporating graph-based features into ML yields robustness against complex communication patterns, which are usually used by attackers to remain undetected.

Though network flows can contain traces of LM, the complex network topologies and diverse applications can introduce noise in flow-based intrusion detection systems. Moreover, widely adopted encryption protocols, such as TLS, further limits the amount of information available from the network traffic, in particular packet payloads. In contrast, the encryption end-point can reveal fine-grained information, such as running processes and user accounts, which enable host-based intrusion detection systems to differentiate between benign and malicious behavior with high precision. Therefore, numerous works leverage host-level information for LM detection.

At the host-level, early detection of LM can be addressed by tracking event logs and identifying malicious events. However, the stealthiness and sparseness of malicious events can render this strategy ineffective. Though, crafting discriminative features for each event can achieve high recall, it comes at a high computational overhead. For instance, Fig. 1 shows that the number of events increase with the network size. Similarly, Fig. 2 shows that feature extraction time for event tagging using the method described in [8] significantly increases with the number of hosts in the network. These drawbacks make this strategy unscalable for very large networks. Furthermore, for complex network infrastructure with sporadic events, tagging individual events can also result in high false positives. In contrast, tagging target assets (TAs) (i.e., hosts targeted by LM) reduce computational overhead. With carefully crafted features from sparse events, it is possible to achieve a high precision in detection performance (see Section IV). Therefore, we focus on the latter strategy for early detection of LM by leveraging host authentication logs.

In this article, we propose a novel approach for ML-based LM detection at the host level. Our main contributions are:

- Leverage ML for identifying TAs to facilitate early detection of LM. We evaluate numerous supervised ML techniques and their ensemble, and compare them in classification performance and overhead.
- Employ graph-based features using real datasets from the Los Alamos National Lab (LANL) [9]. We explore features that are extracted from host authentication logs, and perform feature engineering to reduce dimensionality.
- Due to the highly imbalanced nature of the LANL dataset, we evaluate various over- and under-sampling techniques, and scrutinize their impact on classification performance.
- Compare our approach to state-of-the-art approaches that leverage the LANL dataset for detecting LM. We show that our approach outperforms the other approaches with respect to detection performance and overhead.
- Perturb the attack patterns of LMs and study the impact of varying attack frequency and scale on classification performance. We show that adding synthetic TAs (i.e., noise and variation) in training data helps to mitigate the influence of attack pattern changes, making the system more robust to adversarial attempts.

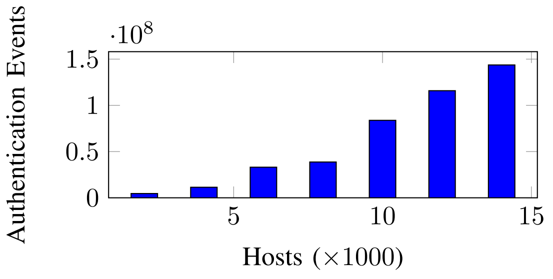


Fig. 1. Number of Hosts vs. Events: Random sample of hosts from the original LANL dataset and corresponding authentication events.

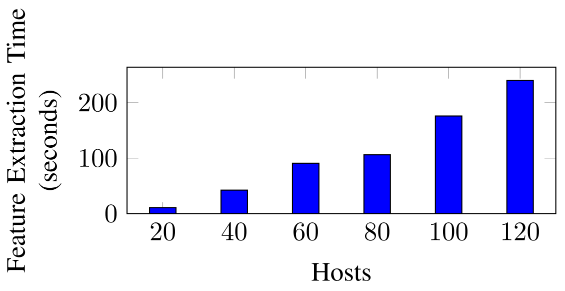


Fig. 2. Number of Hosts vs. Feature Extraction Time: Random sample of hosts from five days in LANL dataset and features from corresponding authentication events.

This article is an extension of our previous work in [10], and evaluates our classifier performance in an adversarial setting. We show the influence of such perturbations on our classifier and instigate its cause, leading to strategies for complementing the training data and address classifier vulnerabilities (see Section IV-C). The rest of this article is organized as follows. Section II provides a background on APT and LM detection, and highlights the most recent related works. We discuss the characteristics of the LANL dataset, delineate the explored sampling algorithms, expose feature extraction and selection, and present the ML techniques employed for TA detection in Section III. In Section IV, we delineate the results of our evaluation and compare with the state-of-the-art approaches for LM detection. We also explore the impact of attack pattern changes on classification performance, and propose a solution to mitigate its influence. We conclude with a brief summary and instigate future research directions in Section V.

II. Background and Related Works

A. Advanced Persistent Threat and Lateral Movement

In an APT, attackers gain access to systems or networks and reside there for an extended period of time. Different from traditional intrusions, APTs have several distinct characteristics: (i) clear objectives, (ii) highly organized and well-resourced, (iii) long-term persistence with repeated attempts, and (iv) evade detection [11]. Due to these characteristics, APTs are challenging to detect. Nevertheless, their distinct characteristics lead to several common attack stages.

Table I summarizes the typical steps performed by an APT. During Reconnaissance, attackers collect information about the target from various resources. They may be sophisticated and gather information from organization websites,

TABLE I
APT STAGES

Step	Summary
Reconnaissance	gather information about the target
Delivery	deliver exploits to the target
Initial Intrusion	get first unauthorized access to the target
Command and Control	control compromised hosts
Lateral Movement	move inside network and expand control
Data Exfiltration	steal sensitive data

multimedia, and social networks. Then by tricking the employees to click on crafted URLs or open malicious attachments, attackers deliver the malware onto the target hosts. With the malware successfully installed on the victim host, a command and control channel is used to control the victim from the master server. After moving laterally and compromising further hosts, attackers exploit the valueable assets through data exfiltration.

LM is used by adversaries to systematically explore the network to access valuable assets. As a crucial stage in APT, LM normally lasts for the longest time. Attackers gain more information about the system by scanning the network and compromising more hosts. To explore the internal network and remain undetected, attackers perform this stage slowly, leaving minimal footprints. They often reduce the attack frequency and utilize legitimate system tools [12]. Mixed in a large volume of benign activities, identifying LMs is challenging.

B. Lateral Movement Detection

Two attributes of LM make its detection non-trivial. First, a plethora of possible attacking techniques (e.g., Pass the Hash, Remote Desktop Protocol, Remote services, *etc.*) can be used during LM [13]. These leave different traces on either the network, the hosts, or both. Second, to remain undetected, attackers launch their attacks slowly and infrequently. As a result, their traces are mixed with the copious benign activities. Typically, researchers tackle the first problem by using anomaly detection. As opposed to detecting states (e.g., number of bytes received, CPU usage, memory usage, *etc.*) caused by the attack, they detect new attacks that result in an undefined state. For the second problem, different methods are used to collect all evidence available, seeking traces of LMs from both the host and the network. They also balance the dataset using well-known or custom methods.

1) Anomaly Detection: Anomaly detection has been extensively used for network intrusion detection [14]. As opposed to methods targeting characteristics of individual attacks, it detects activities aberrant from normal forms, which makes the method robust to unknown attacks. After building the knowledge of the normal behavior, the system assigns abnormal scores to new activities, which are then classified accordingly. Research efforts in this area can be classified into statistic-based, knowledge-based and ML-based anomaly detection.

Statistic-based anomaly detection systems (SB-ADS) build the baseline from historical data, such as the frequency of network events, kind of protocols used, and the number of destination hosts contacted. By comparing the profiles of historical data and that of new incoming events, SB-ADS decides the degree of irregularity with a score. Over a span of time, statistic-based systems can efficiently detect malicious activities with well-defined normal profile. However, they are vulnerable when the normal profile is contaminated. Attackers can interfere during the training phase, which can cause attacking events to be classified as benign. On the other hand, knowledge-based ADS infers the legitimacy of events according to pre-defined rules. However, benign patterns that have not been anticipated, can be misclassified in this approach. Besides, composing the rules is difficult and time consuming [14]. Compared to the previous two methods, ML-based approach can adapt to behavioral changes. It resembles SB-ADS in theory, but with new labeled data and properly selected models, ML-based approach can improve its performance progressively. In this article, we focus on leveraging ML-based approaches to capitalize on their robustness to new attack patterns.

2) Host-Based LM Detection: Liu *et al.* [15] propose a ranking system to detect targeted hosts. The authors first construct a connection graph from Windows security events generated by Kerberos service ticket requests. Then they rank the hosts according to path-rate score, which reflects the rarity of a path in the network. To reduce the false positives, the authors build a remote file execution detector and filter out benign behaviours. Their system can detect all LM-related remote file execution. Chen *et al.* [16] leverage features from multiple data sources to identify LM. They utilize rudimentary graph-based features based on host communication, while employing autoencoder to improve feature extraction. To address imbalance in the LANL dataset, the authors propose a custom under-sampling technique. They employ k-nearest neighbor (k-NN) and achieve an average of 91.3% precision in LM detection. However, their evaluation is limited to the k-NN classifier.

Bohara *et al.* [17] propose an unsupervised approach to detect LM. They employ the LANL dataset and inject synthetic attacks into the original dataset, instead of using the redteam events in the dataset. However, these simulated attacks may not capture the true characteristics of real attacks in enterprize networks. Their LM activity simulation follows the susceptible-infected-susceptible virus spread model [18]. The authors extract features from host communication graphs, while principal component analysis (PCA) is used to correlate different features. For detection, they propose a combination of two different detectors to enhance performance. The first detector uses PCA and k-Means, while the second employs PCA and extreme value analysis. This combination achieves a 88.7% true positive rate.

Chawla *et al.* [19] apply Convolutional Neural Network (CNN) and Gated Recurrent Unit (GRU), a gating mechanism in Recurrent Neural Network (RNN), on the ADFA-LD [20] dataset. The CNN layers can capture local correlation of structures in the sequences and execute in parallel to improve performance, while the RNN with GRU can learn sequential correlations from higher-level features. Their system can reach an Area Under the Curve (AUC) score of 0.81 after training on normal sequences. However, they only employ a well balanced dataset, which does not depict a true attack scenario. In reality, APTs are stealthy, which will result in a highly unbalanced trace, as evident in the LANL dataset. Failing to test against imbalanced data undermines their classifier's ability against real attacks.

Tuor *et al.* [21] and Brown *et al.* [22] propose RNN for log-level intrusion detection. Tuor *et al.* introduce a language modeling framework for generic log-level intrusion detection, while Brown *et al.* extend a previous framework and focus on developing RNN models with attention mechanism. These efforts do not employ feature engineering, but rather the ML models directly leverage tokenized log lines. They achieve AUC of 0.98 and 0.99, respectively. However, AUC is impacted when the dataset is highly imbalanced. In contrast, our approach operates on the host-level, whereas the aforementioned approaches detect on the log-level.

Kaiafas *et al.* [8] construct a bipartite graph, an inspiration for our approach, to extract graph-based features and employ an ensemble of ML models to improve classification performance on the LANL dataset. However, the authors only perform *k*-fold cross-validation and do not evaluate the robustness of their ML models to unseen data. This is crucial to ensure the detection of zero-day APTs. We highlight this limitation in Section IV.

C. Adversarial Learning

ML has been extensively explored for cybersecurity. However, research shows that crafted adversarial samples can hamper the performance of the model [23]–[26]. Xu *et al.* [26] propose a generic method to identify evasive samples. After the test against two recent PDF malware detectors, their system successfully evades detection with 100% success rate. Biggio *et al.* [27] experiment with a gradient-based approach to evade detection and show that popular classification algorithms, such as support vector machine (SVM) and neural networks are vulnerable. Anderson *et al.* [28] leverage generative adversarial networks to thwart the performance of a deep learning-based detector, which detects the use of Domain Generation Algorithms. We perform experiments to explore the influence of adversarial samples on our approach for TA detection (see Section IV-C).

III. Methodology

A. Dataset

1) Characteristics: The LANL dataset contains logs from multiple data sources, including authentication and flow logs. In our previous work [10], we highlight the lackluster performance of flow-based features, which can be attributed to the inferior quality of flow data. This undermines the suitability of flow-based features to detect TAs during LM in the LANL dataset. Therefore, in this work we only explore the authentication log.

The authentication log is composed of over 450 million authentication events from Windows-based desktop computers, spanning 58 days. Among these events there are 749 redteam-driven malicious events, distributed in the first 30 days of the dataset, as depicted in Fig. 3. We leverage data in this time frame, which consists of about 230 million events from 14,582 benign hosts and 299 hosts associated with redteam activities. However, the malicious activities are a small fraction of the overall activities in the LANL dataset. Fig. 4 shows the daily events distribution. In comparison to Fig. 3, it also reveals that malicious activities are rare, both in general and on a daily basis.

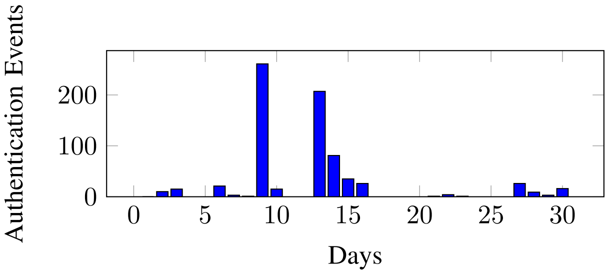


Fig. 3. Redteam activities distribution.

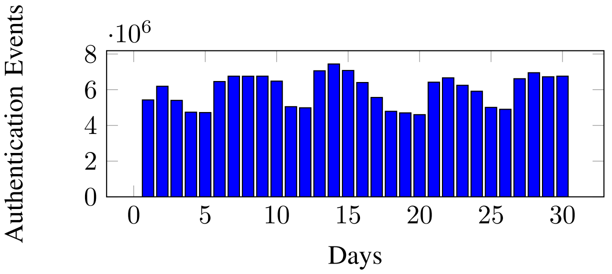


Fig. 4. Authentication events distribution.

We do not consider local redteam authentication events, i.e., malicious events where the source and destination hosts are the same. The behavior of an attacker that performs malicious activity within a physical machine tends to be quite different. Such an attacker has access to the physical interfaces of the host, hence their attack strategy and behaviour can be very sophisticated. Evaluating such behavior is out of the scope of this article. Nevertheless, we capitalize on the number of infrequent events. In total, there are 8,941 hosts involved in 41,400 authentication events that occur only once in the dataset. Out of the 295 TAs, 280 are involved in such events. Therefore, considering the event infrequency, i.e., *sparseness*, which is a trait of the majority of the malicious authentication events, as depicted in Fig. 6, can potentially facilitate the detection of TAs.

2) Balancing: Sampling algorithms are employed when a dataset is highly imbalanced. An imbalanced dataset can result in a classifier that is biased towards the majority class, due to the nature of the training procedure. The sampling algorithms used to alleviate this issue can be classified into two categories, under-sampling and over-sampling. While under-sampling approaches balance the dataset by reducing the data points in the majority class, the over-sampling approaches increase the data points in the minority class. Therefore, the under-sampling algorithms are known to inherently lose critical information, while the over-sampling algorithms suffer from over-fitting [29]. However, a potential advantage of under-sampling is the reduced computational overhead. On the

TABLE II
SIGNIFICANT FEATURES EXTRACTED FROM AUTHENTICATION LOGS (SEE SECTION IV-B1)

Feature	Definition
$ID_{usr}(dst_j)$	The count of unique *username* used to logon to dst_j
$ID_{src}(dst_j)$	The count of unique *source* hosts that logon to dst_j
$ID_{(usr,src)}(dst_j)$	The count of unique *(username, source)* pairs that logon to dst_j
$IDAF_{usr}(dst_j)$	The average over all *username* of $AVG_{dst_j}(username)$, where $AVG_{dst_j}(username)$ is the number of times *username* is used to logon to dst_j divided by the number of days *username* used to logon to dst_j
$IDAFSTD_{usr}(dst_j)$	Standard deviation of $AVG_{dst_j}(username)$
$IDS_{usr}(dst_j)$	The sum over all *username* of $SF(*, username, dst_j, \theta, \beta)$, where SF is exposed in Algorithm 1
$IDS_{src}(dst_j)$	The sum over all *source* of $SF(source, *, dst_j, \theta, \beta)$
$IDS_{(usr,src)}(dst_j)$	The sum over all *(username, source)* pairs of $SF(source, username, dst_j, \theta, \beta)$
$WIDS_{(usr,src)}(dst_j)$	The sum over all *(username, source)* pairs of $SF(source, username, dst_j, \theta, \beta)$ weighted by $ODS_{(usr,dst)}(source)$
$ODS_{usr}(src_i)$	The sum over all *username* of $SF(src_i, username, *, \theta, \beta)$
$ODS_{dst}(src_i)$	The sum over all *destination* of $SF(src_i, *, destination, \theta, \beta)$
$ODS_{(usr,dst)}(src_i)$	The sum over all *(username, destination)* pairs of $SF(src_i, username, destination, \theta, \beta)$
$ODAFSTD_{(usr,dst)}(src_i)$	Standard deviation of $AVG_{src_i}(username,\ destination)$, where $AVG_{src_i}(username, destination)$ is the number of times *username* is used by src_i to logon to $destination$ divided by the number of days *username* is used by src_i to logon to $destination$
$ODAFSTD_{usr}(src_i)$	The standard deviation of $AVG_{src_i}(username)$, where $AVG_{src_i}(username)$ is the number of times *username* is used in a remote login attempt initiated by src_i divided by the number of days *username* is used by src_i in a remote login attempt
$MSF(dst_j)$	The maximum over all src_i of $ODS_{(usr,dst)}(src_i)$ where $SF(dst_j, src_i, in, \theta, \beta) > 0$
$SUR(dst_j)$	The number of unique *username* used to sparsely logon to dst_j (*i.e.*, $SF(dst_j, (username, source), in, \theta, \beta) > 0$ for at least half the logon events $(username, source, dst_j)$) divided by the number of unique *username* used to logon to dst_j
$AS(Host_j)$	$MSF(Host_j) * SUR(Host_j)$

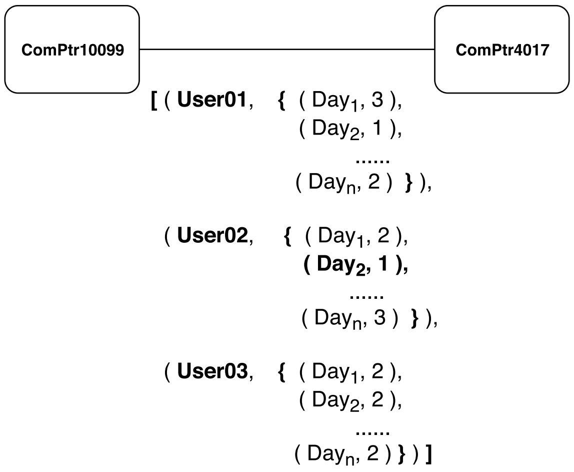


Fig. 5. Graph representation of authentication events.

other hand, some classifiers have the capability to overcome the over-fitting due to over-sampling.

We explore different algorithms from both categories for balancing the LANL dataset. The first algorithm is Random Under-Sampling (RUS), which randomly removes samples from the majority class. The second algorithm is Condensed Nearest Neighbour (ConNN), an under-sampling algorithm based on *k*-NN [30]. This algorithm keeps all samples in the minority class and uses 1-NN classifier to determine whether to retain the data point in the majority class or not. The next algorithm is Repeated Edited Nearest Neighbours (RENN), which implements multiple iterations of Edited Nearest Neighbours (ENN) [31]. For the over-sampling algorithms, we start with random over-sampling (ROS), followed by the well-known Synthetic Minority Over-sampling Technique (SMOTE) [32]. SMOTE over-samples data points by creating their synthetic counterparts. This is achieved by computing a vector between a data point and one of its neighbours. Another over-sampling algorithm is the Adaptive Synthetic (ADASYN) [33]. ADASYN also leverages *k*-NN to adaptively generate synthetic data.

We employ the above sampling algorithms after feature extraction. This is primarily because applying them directly on the authentication log can sabotage the extraction of graph-based features. For example, all authentication events pertaining to a particular username may get eliminated due to under-sampling. Similarly, over-sampling without considering the diversity of hosts in the dataset may result in emphasizing a single type of host. We study the influence of these sampling algorithms on TA detection in Section IV.

B. Feature Extraction

We extract a total of 29 features from the authentication log, more precisely from a graph representation of the authentication events. As the features are primarily based on the in- and out-degree of different hosts, we build an authentication graph that is efficient for frequent reference. We first start by building the authentication graph $G = (U, V, E)$, where U represents the hosts that appear as sources in the authentication log, while V represents the hosts that appear as destinations. Edges in E link pairs $(u, v) \in U \times V$ and summarize all authentication events involving source u and destination v.

Authentication events are inserted in the graph as shown in Fig. 5. For example, consider an authentication event $e(Day_2, User02, ComPtr10099, ComPtr4017)$, where Day_2 represents the day when the logon was recorded, *User02* is the username used in the logon attempt, and *ComPtr10099* is the source host used by *User02* to logon to destination host *ComPtr4017*. Assuming that *User02* was already recorded

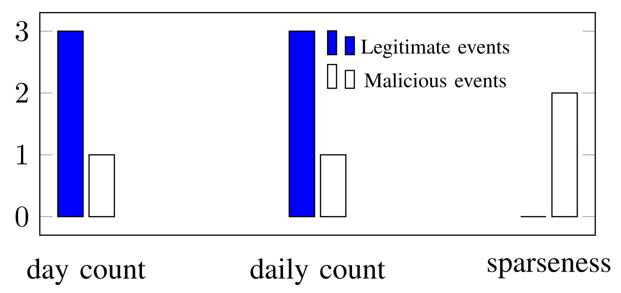


Fig. 6. Median day count, per-day count and sparseness (see Algorithm 1) of malicious and legitimate authentication events.

logging into *ComPtr4017* from *ComPtr10099* twice on Day_1, and 3 times on Day_n, the event *e* is added to the edge linking *ComPtr10099* to *ComPtr4017* on the graph *G* with a count of 1 (i.e., *e* is the first and unique event linking *User02*, *ComPtr10099*, and *ComPtr4017* on Day_2), as depicted in Fig. 5. Once the graph *G* is completed, we build dictionaries that are used to extract the features, as described in [34].

A complete list of 29 authentication-based features is available in [34], while Table II delineates the 17 features after feature selection, as described in Section IV-B1. A high-level description of the authentication-based features is provided below.

a) In-degree (ID) and out-degree (OD): In the early phase of LM, attackers use stolen credentials to attempt logging into and eventually compromising other hosts. This will result in the increase of ID of the targeted hosts and OD of successfully compromised ones.

b) In-degree-avg-frequency (IDAF) and out-degree-avg-frequency (ODAF): Infrequent malicious authentication events will have little impact on ID/OD in the presence of a much larger number of benign authentication events. Thus, they can be overlooked by the classifier. We consider IDAF, the daily average number of authentication events targeting the host, as well as ODAF, the average number of authentication events originating from the host, and leverage the discriminatory nature of these features.

c) IDAF-standard-deviation (IDAFSTD) and ODAF-standard-deviation (ODAFSTD): Sparse malicious authentication logs can be shadowed by regular and repetitive benign logons when calculating IDAF and ODAF. On the other hand, the standard deviation of IDAF will be higher for TAs targeted by a mix of frequent legitimate logons and sparse malicious logons, than non-TAs. Similarly, compromised hosts will have higher ODAFSTD than benign ones.

d) In-degree-sparseness (IDS) and out-degree-sparseness (ODS): In order to capture infrequent events that are likely to be malicious, we introduce a *sparseness function* (SF), as depicted in Algorithm 1. SF considers infrequent events with a specific (combination of) *source host*, *destination host*, or *username*, and assigns a higher score to more infrequent events. This amplifies the impact of such events on graph-based features, which are otherwise largely affected by benign events. IDS and ODS reflect the sparseness of the incoming and outgoing logons, respectively. In this case, SF evaluates the sparseness of these events and amplifies the impact of sparse authentication events when computing the ID of a TA and OD

Algorithm 1 Sparseness Function (SF)

```
input : Source host $Src$, username $Usr$, destination host $Dst$,
        threshold $\theta$, factor $\beta$
output: $Sparseness$, a sparseness score of event defined by
        $Src$, $Usr$, and $Dst$
1: Initialize $Events$ to all events in authentication log
2: $Sparseness \leftarrow 0$
   /* filter(*) is a no-op, $countByDays()$ counts the number
   of days where the event occurs */
3: $TotalDays \leftarrow$
4:     $Events.filter(Src, Usr, Dst).countByDays()$
5: if $TotalDays \leq \theta$ then
6:     $Sparseness \leftarrow max(TotalDays * \beta -$
7:             $Events.filter(Src, Usr, Dst).count(), 0)$
       /* $\theta$ is the number of days an event has to occur before
       it is qualified as non-sparse */
       /* $\beta$ is the number of times an event has to occur daily
       before it is qualified as non-sparse */
8: end if
9: return $Sparseness$
```

of a compromised host. As sparse malicious logons receive higher SF scores, TAs are expected to have higher IDS than non-TAs, and compromised hosts to have higher ODS than benign ones.

e) Weighted-in-degree-sparseness (WIDS): To distinguish between TAs and non-TAs with comparable IDS but legitimately targeted by a higher number of logons (e.g., servers), we weigh the sparseness of incoming logons by the ODS of the source.

f) Maximum-sparseness-factor (MSF) and suspicious-user-rate (SUR): A common characteristic of TAs is that they have been occasionally logged into with malicious intent. The MSF of a particular host denotes the ODS of the source that is most likely to be malicious and that has sparsely logged into that host. The higher is the MSF of a host the more likely it is a TA. The SUR of a given host is the proportion of usernames used to sparsely log into the host.

g) Attack score (AS): The AS of a host reflects the likelihood of it being a TA. The higher the AS of a host, the more like it is a TA. AS is the product of MSF and SUR, hence it is correlated with MSF and SUR. Experiments with AS show a promising boost with respect to precision and recall. As the product of MSF and SUR, AS serves as an evident sign that the host has been tempted by an actively probing source host. It further reveals that selected ML models can not capture the product relationship of different features. With all the features, the classifier can better distinguish the boundary between TAs and non-TAs.

C. ML Techniques

With graph-based features extracted, we evaluate several ML techniques to detect TAs during LM. We start with decision tree (DT), a non-parametric supervised learning method. We also leverage random forest (RF), which is a classifier that uses multiple DTs to improve classification performance and avoid over-fitting. LogitBoost (LB) is another learning algorithm based on DT that we leverage in our evaluation. We also assess logistic regression (LR), which is very efficient and does not require feature scaling. However, its performance deteriorates with highly correlated features. We evaluate the above ML algorithms along with other well known algorithms, such as SVM, *k*-NN, and gaussian naïve bayes. However, we omit discussion on the latter due to their inferior performance in our evaluation.

D. Evaluation Metrics

In order to measure the performance of ML models, we use a variety of metrics. These include:

$$Precision = \frac{\text{TruePositive}}{\text{TruePositive} + \text{FalsePositive}} \times 100\%$$

$$Recall = \frac{\text{TruePositive}}{\text{TruePositive} + \text{FalseNegative}} \times 100\%$$

$$F1\ score = 2 \times \frac{\text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}}$$

The precision and recall are better criterion compared to accuracy, false positive rate, and true negative rate, to assess the performance of a classifier when the dataset is imbalanced. The F1 score is essentially a harmonic mean of precision and recall, which represents the overall performance of a classifier. A higher F1 score indicates both low false positives and low false negatives (i.e., true TAs are identified without raising many false alarms). In addition, we plot receiver operating characteristic (ROC) curve to illustrate the performance of a classifier at different classification thresholds. We also calculate AUC to quantify ML performance.

IV. Evaluation

A. Environment

1) Hardware: We perform data pre-processing and analysis on a cluster of four nodes, each of which has a Intel Xeon 3.30GHZ CPU and 16GB RAM. These nodes are interconnected using 10Gbps Ethernet. ML model training, validation and testing is performed on a machine equipped with 2 x Intel Xeon 2.20GHz CPU and 384 GB RAM.

2) Software: We leverage Numpy [35], Scipy [36], and Pandas [37] for data pre-processing. Imbalanced-learn [38] is employed for balancing the training datasets, while Scikit-learn [39] is used for building ML models.

B. TA Detection

1) Feature Selection: We start by evaluating the performance of different ML classifiers with graph-based features extracted from authentication events. Table III showcases the result of *k*-fold cross-validation ($k = 10$) using all 29 features [34] extracted from the first 30 days in the LANL dataset. We choose θ and β based on trial-and-error and the frequency of benign activities in the dataset. Most authentication events for a given combination of (*src*, *username*, *dst*) occur for more than three times per day and exist over three days. Hence, we set $\theta = \beta = 3$. The parameters for the ML techniques are set based on their performance,

TABLE III
ML PERFORMANCE USING ALL AUTHENTICATION-BASED FEATURE SET (29 FEATURES)

ML model	Precision	Recall	F1 score	Training time (s)
DT	75.26%	75.77%	0.75	0.11
RF	81.36%	80.12%	0.81	3.11
LB	79.64%	79.76%	0.79	5.46
LR	61.31%	53.56%	0.52	6.13

TABLE IV
PEARSON CORRELATION MATRIX FOR MOST CORRELATED AUTHENTICATION-BASED FEATURES

FID	2	3	10	11	15	18	22	23	24	26	27
16	0.9	0.9	0.9	0.9	0.9	0.0	0.0	0.0	0.0	0.0	0.0
20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.6	0.9	0.0	0.8
25	0.0	0.0	0.0	0.0	0.0	0.4	0.6	0.0	0.0	0.1	0.0
29	0.0	0.0	0.0	0.0	0.0	0.6	0.4	0.0	0.0	0.6	0.0

TABLE V
ML PERFORMANCE ON REDUCED AUTHENTICATION-BASED FEATURE SET (17 FEATURES)

ML model	Precision	Recall	F1 score	Training time (s)
DT	77.59%	75.45%	0.76	0.05
RF	83.72%	81.23%	0.82	2.06
LB	78.99%	80.25%	0.80	2.26
LR	68.13%	54.55%	0.60	5.99

i.e., we choose the parameters that exhibit the best result in TA detection. DT is set to a maximum depth of 6, while RF uses 400 estimators with a maximum depth of 12. LB uses 100 estimators and a DT regressor with a maximum depth of 3. LR uses a tolerance of 0.0001 and a regularization strength of 1.

With the exception of LR, which performs poorly, the other ML techniques classify TAs with relatively high precision and recall (i.e., over 75%). RF outperforms DT and LB with the highest F1 score, while saving about *14s* in feature extraction time. However, the number of ML features not only influence the computational overhead, but can also result in model over-fitting. Therefore, we study the correlation between the features to further reduce their number. Table IV shows that among the 29 features, 11 (column features) are correlated with 4 others (row features), with a Pearson coefficient exceeding 0.6. Hence, we remove all the 11 correlated features from the feature set. Many of these features report on the daily average logon times; per host, per user, and per (host, user) combination. These features are too generic and fail to describe the true nature of LM, which makes them less discriminative in tagging TAs. We further remove the *out-degree-avg-frequency* feature, which is only significant if there is evidence that the TA is compromised and is actively attempting to move laterally. However, this is not evident in the LANL dataset. We re-evaluate the ML techniques after removing the 12 aforementioned features. The authentication-based Feature IDs (FIDs) are available in [34].

As depicted in Table V, the F1 score for all ML techniques improve with RF outperforming all other classifiers. Furthermore, LR shows the highest improvement in TA detection with an F1 score increase of 8%. Even though DT and

TABLE VI
ENSEMBLE LEARNING USING MAJORITY VOTING

Ensemble	Precision	Recall	F1 score	Training time (s)
RF, LB, DT	80%	80.67%	0.80	2.60

TABLE VII
ENSEMBLE LEARNING USING WEIGHTED VOTING, PRIORITIZING STAND-ALONE ML PERFORMANCE

RF	LB	DT	Precision	Recall	F1 score	Training time (s)
$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	80.00%	80.41%	0.80	2.86
$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{4}$	80.70%	80.89%	0.81	2.78
$\frac{1}{2}$	$\frac{1}{3}$	$\frac{1}{6}$	80.68%	80.97%	0.81	2.60

LB are immune to highly correlated features [40], we notice a slight increase in their performance. The removed features primarily pertain to standard deviation and out-degree. On a single host, different users can have distinct authentication patterns, which will result in high values for standard deviation-based features, causing confusion for the classifiers. Furthermore, if TAs do not have an exploring behaviour, the out-degree-based features can also degrade the classifier performance. Hence, in the following experiments we use the reduced feature set of 17 authentication-based features (see Table II in Section III-B).

2) Ensemble Learning: To improve the performance of the stand-alone ML models, we consolidate them using ensemble learning. Due to the lackluster performance of LR in comparison to other ML models (see Table V), we remove it from the list of potential classifiers in the ensemble approach. First, we employ the majority voting (MV) algorithm [41] that leverages all ML models in the ensemble in a uniform manner, and use k-fold cross-validation ($k = 10$) on the first 30 days of the LANL dataset. The detection of TAs during LM using MV over RF, LB and DT is shown in Table VI. This results in an inferior performance to stand-alone RF, since low performing classifiers can influence the voting process.

Evidently, MV is unable to boost the performance of the best stand-alone classifier. Therefore, we explore another ensemble approach, namely weighted voting (WV) [42], where we can assign weights to ML models based on their stand-alone performance. Intuitively, this can identify a higher number of true positives (i.e., TAs during LM) that are missed by RF, the best performing stand-alone classifier. However, with multiple combination of weights assigned to the ML models in Table VII, stand-alone RF still outperforms WV. This reveals that at the classification boundary where RF is unable to differentiate between TAs and non-TAs, LB and DT also suffer and do not facilitate better performance. Besides, these ensemble approaches increase training time, which undermines their suitability for early LM detection. Therefore, we choose the stand-alone RF classifier as *our model* for further experiments.

3) Balancing the Dataset: The distribution of the dataset can severely influence the performance of classifiers. For example, in the case of an imbalanced dataset (e.g., sparse malicious host events versus benign events), ML models are more likely to classify new data to the majority class.

TABLE VIII
OVER-SAMPLING WITH DIFFERENT ALGORITHMS USING STAND-ALONE RF (17 FEATURES)

Algorithm	TA/non-TA	Precision	Recall	F1 score	ST (s)	TT (s)
ROS	0.02	62.34%	95.36%	0.7539	0.01	4.27
SMOTE	0.02	61.99%	95.71%	0.7524	0.01	4.62
ADASYN	0.02	62.08%	96.07%	0.7542	0.06	4.66

Though, balancing the dataset can alleviate this issue, it may also sabotage the performance of the classifier by impacting graph-based features.

We evaluate the robustness of our ML model by training and testing it on logs recorded on different days. Redteam activities are only conducted on certain days, generating malicious events that account for a very small fraction of the total number of authentication events (i.e., less than 0.0001%). Therefore, we reserve day 9, the day with the highest number of malicious authentication events, for testing, while the remaining days are chosen for training the model. We evaluate several well-known sampling algorithms (see Section III) to balance the training dataset. For each sampling algorithm, we use distinct seeds across 5 iterations and compute the average for each metric. These seeds are consistent across the sampling algorithms. Furthermore, each sampling algorithm has its own best sampling rate, i.e., the ratio of TA versus non-TA (TA/non-TA). Hence, we experiment with different sampling rates and select the best sampling rate to portray the corresponding results.

a) Over-sampling: The comparison of three different over-sampling algorithms, namely ROS, SMOTE, and ADASYN, is highlighted in Table VIII. SMOTE results in the second highest recall, as synthesizing minority points help in stressing the TA class. However, the randomness in synthetic points does not capture the true nature of original TAs, resulting in a lower precision. In contrast, ADASYN achieves better precision and recall. It generates synthetic points closer to the decision boundary, thus enabling the classifier to better distinguish TAs from non-TAs. As opposed to over-sampling only a portion of the minority points, ROS simply replicates TAs. Uniformly stressing on all TAs preserves the authenticity of the TA class and its behavior to a large extent in comparison to synthesizing, thus resulting in the highest precision. Each algorithm over-samples the dataset with the same sampling rate, thus the training time (TT) is similar. In contrast to ADASYN, ROS and SMOTE consume less sampling time (ST) due to their simpler sampling mechanism.

b) Under-sampling: Recall that RUS randomly removes samples from the majority class. This may result in a high number of non-TAs (i.e., majority class) that have similar traits as certain class of TAs, negatively impacting precision. This is evident in the lower precision of RUS in comparison to ConNN, as shown in Table IX. But the TAs that starkly differ from non-TAs are still classified with high recall. A similar affect can be seen with RENN, which removes non-TAs that are not very similar to their neighbors. In contrast, ConNN preserves the non-TAs that are different from their neighbors. Therefore, under-sampling with ConNN results in the best F1

TABLE IX
UNDER-SAMPLING WITH DIFFERENT ALGORITHMS USING STAND-ALONE RF (17 FEATURES)

Algorithm	TA/non-TA	Precision	Recall	F1 score	ST (s)	TT (s)
RUS	0.02	60.9%	96.55%	0.747	0.01	2.45
ConNN	0.51	62.47%	95.12%	0.754	130.88	0.99
RENN	0.01	60.07%	97.62%	0.743	2.81	3.32

TABLE X
COMPARING DIFFERENT ALGORITHMS USING STAND-ALONE RF (17 FEATURES)

Algorithm	TA/non-TAs	Precision	Recall	F1 score	ST (s)	TT (s)
ADASYN	0.02	62.08%	96.07%	0.7542	0.06	4.66
ConNN	0.51	62.47%	95.12%	0.754	130.88	0.99
Unbalanced	0.01	61.54%	95.23%	0.75	0	3.61

score with precision and recall of 95.12% and 62.47%, respectively. Due to its simplicity, RUS incurs the least sampling time. In contrast, ConNN suffers from the highest sampling time, but it also reduces the number of non-TAs to the largest extent, which positively impacts training time.

c) Comparison: We highlight the over- and under-sampling algorithms with the highest F1 score in Table X, along with no sampling (i.e., imbalanced training dataset). As evident, ADASYN increases the precision and recall by 0.54% and 0.84%, respectively. However, this comes at the cost of increased sampling and training times, which undermines its suitability. On the other hand, ConNN increases precision by 0.93%. However, its sampling time is very high, while the F1 score is comparable without any sampling. Therefore, to avoid this overhead, we proceed without any sampling to detect TAs during LM in the LANL dataset.

4) Comparative Analysis: To further evaluate our approach, we compare our model with two state-of-the-art approaches for LM detection. We implement the approaches in Chen *et al.* [16] and Kaiafas *et al.* [8]. To achieve a fair comparison, we balance the dataset according to the algorithm in [16], which preserves the redteam events while under-sampling the benign activities. Due to scalability issues in [8], we only leverage data for k-fold cross-validation ($k = 10$) from day 9. As depicted in Table XI, our model outperforms Chen *et al.* in precision, recall and F1 score. However, our approach consumes more feature extraction time (FET) and model training time.

Kaiafas *et al.* marginally outperforms our model in recall, with an improvement of 0.02 in F1 score. However, their feature extraction and model training times are magnitudes higher than both Chen *et al.* and our approach. In the balanced dataset, there are about 97,000 authentication events and their overhead is largely due to feature extraction for each individual event. In contrast, our approach strikes a balance between performance and overhead.

Following cross-validation, we evaluate the robustness of the aforementioned approaches on never seen data. Thus, we leverage authentication events from day 9 as the test dataset, while the remainder of the dataset (i.e., 29 days) is used for training. In this case, the training dataset is composed

TABLE XI
TA DETECTION USING STAND-ALONE RF AND CROSS-VALIDATION VERSUS [8], [16]

Classifier	Precision	Recall	F1 score	FET (s)	TT (s)
Our Model	97.02%	93.04%	0.95	169.35	1.45
Chen *et al.*	7.24%	73.12%	0.13	0.69	5.29
Kaiafas *et al.*	100%	93.47%	0.97	100.81	23332.37

TABLE XII
ROBUSTNESS OF TA DETECTION USING STAND-ALONE RF VERSUS [8], [16]

Classifier	Precision	Recall	F1 score	FET (s)	TT (s)
Our Model	60.58%	98.81%	0.75	475.46	2.56
Chen *et al.*	2.98%	3.16%	0.030	823.51	59.45
Kaiafas *et al.*	—	—	—	> 360000	—

TABLE XIII
ROBUSTNESS OF TA DETECTION USING STAND-ALONE RF VERSUS [8], [16] ON REDUCED TRAINING DATASET

Classifier	Precision	Recall	F1 score	FET (s)	TT (s)
Our Model	61.24%	94.05%	0.74	475.46	2.56
Chen *et al.*	4.64%	9.52%	0.06	11.22	0.66
Kaiafas *et al.*	9.58%	45.83%	0.16	40488.56	1903.24

of over 220 million log entries, which can potentially introduce a lot of noise. As depicted in Table XII, the model from Chen *et al.* fails miserably with a near-zero recall when tested on never seen data. In [16] the feature set consists of generic statistical features that fail to distinguish TAs in a noisy environment. Unfortunately, we are unable to extract features for Kaiafas *et al.* for this robustness evaluation in a reasonable amount of time. Thus, the robustness evaluation for their model is unavailable. In contrast, our model shows remarkable performance with a recall and F1 score of 98.81% and 0.75, respectively.

Nevertheless, to compare the robustness of Kaiafas *et al.* we reduce the cardinality of the training dataset from the previous experiment. Data from days 13, 14, and 15 is used for training, while day 9 is reserved for testing. For a fair comparison, we leverage the under-sampling method from Chen *et al.* for both comparative models. Note that Kaiafas *et al.* do not expose their sampling approach in detail. Furthermore, no sampling is applied to our model. As shown in Table XIII, our model significantly outperforms other approaches. Even though Kaiafas's model performs quite well in cross-validation, it fails in robustness to unknown TAs.

The features in [8] fail to differentiate between TAs and non-TAs in large networks. Diversity in authentication events can result in TAs and non-TAs having similar values with regard to less crafted features, such as the number of successful/failed authentication events. Such noise will influence the performance of ML models that leverage less-thought-of features. However, with features based on the degree of sparse events, our model is able to filter out noise and differentiate TAs from non-TAs. Fig. 7 shows the ROC curve, which indicates that our model has the highest AUC of 0.995. Note that Kaiafas's model is using MV, which is not feasible for plotting as a ROC curve.

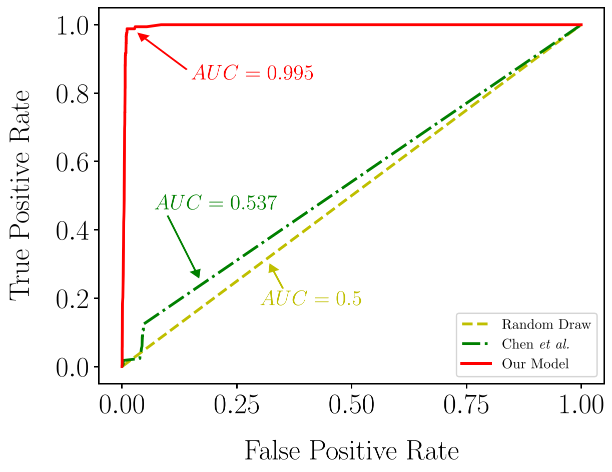


Fig. 7. ROC curve for robustness in TA detection using stand-alone RF versus [8], [16], with days 13, 14 and 15 for training, and day 9 reserved for testing.

In comparison to previous robustness result, our model shows a marginal loss in precision and recall. However, our model out performs both state-of-the-art approaches, with a high recall of over 94%, while the F1 score is the highest at 0.74. Note that the F1 score of 0.74 is primarily attributed to low precision (i.e., high false alarms) in the robustness analysis. Without undermining the importance of false alarms, we believe that false negatives are significantly more serious, as they can compromise the integrity of an enterprise network. This is when a malicious activity is tagged as legitimate. Hence, our primary objective is to maximize recall.

C. Adversarial Learning

In APT, attackers during LM strive to introduce minimal influence. Highly skilled attackers can trick a system by adding small randomness in their activities or imitating benign behavior after long-term scouting. However, in practice, it is very difficult for an attacker to collect behaviour information of benign users, as it may require administrative privileges, and access to system drivers and administrative tools. This will result in evident traces, alerting the administrators of abnormal activity, e.g., abnormal memory usage and degradation in system performance. Therefore, we study the influence of adversarial attempts for the former scenario.

We assume that the attacker cannot access behaviour profile of benign users, the model or the training data. Therefore, the adversary introduces variations in attack patterns to potentially trick the classifier. Since our approach relies on authentication patterns, variations in other aspects, such as system vulnerability and perturbing tools, will not influence the classifier's performance. Our approach capitalizes on the sparsity of malicious authentication events. Thus, we evaluate the influence of an adversary perturbing its probing pattern. More specifically, this includes perturbing the number of TAs and the frequency of malicious authentication events, as shown in Fig. 8. An attacker can both increase the attacking frequency (in b) or decrease it. Similarly, an attacker can also increase the number of TAs (in c) or reduce it. We start by exploring the impact of perturbations in both of these aspects.

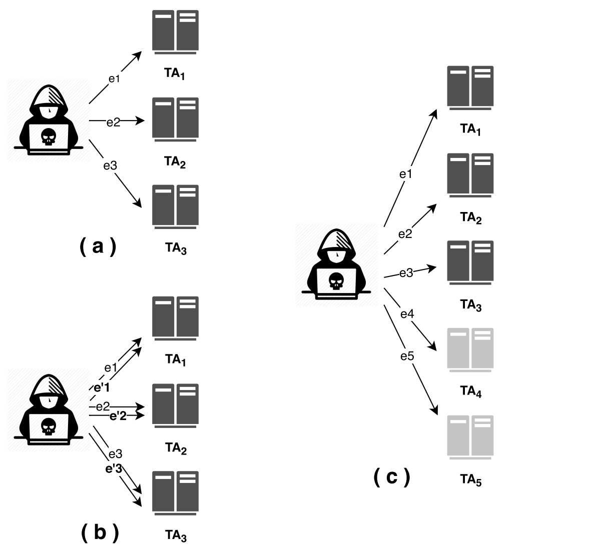


Fig. 8. Attacker changes probing pattern: (a) the original attack, (b) increasing attack frequency (c) increasing the attacking scale (i.e., attack more TAs). Note that arrow labels correspond to authentication events.

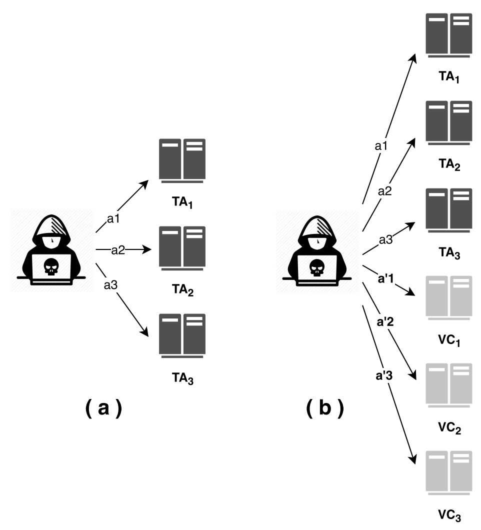


Fig. 9. Example of increase in number of TAs with $Ratio = 2$: (a) original attack, (b) attacking more TAs, for each new host (i.e., victim VC_i for corresponding original TA_i) map to it the original attack a_i on TA_i, i.e., a_i'. Note, attack a_i on TA_i is a set of authentication events.

1) Perturbing Number of TAs: To simulate attackers probing different number of TAs, we first define the *Ratio* between the sampled TAs and the original TAs as follows:

$$Ratio = \frac{\text{Number of TAs after sampling}}{\text{Number of TAs in the original dataset}}$$

For *Ratio* greater than 1 (i.e., increase in number of TAs), we randomly select (*Ratio* -1) $\times$ *Original Number of TAs* from within non-TAs. Then for each of the chosen new host (i.e., victim), we simulate the attack from the original redteam in a one-to-one manner. An example is illustrated in Fig. 9. Similarly, for *Ratio* smaller than 1 (i.e., decrease in number of TAs), we randomly select *Ratio*$\times$ *Original Number of TAs* from the redteam, and only preserve redteam events related to the selected TAs.

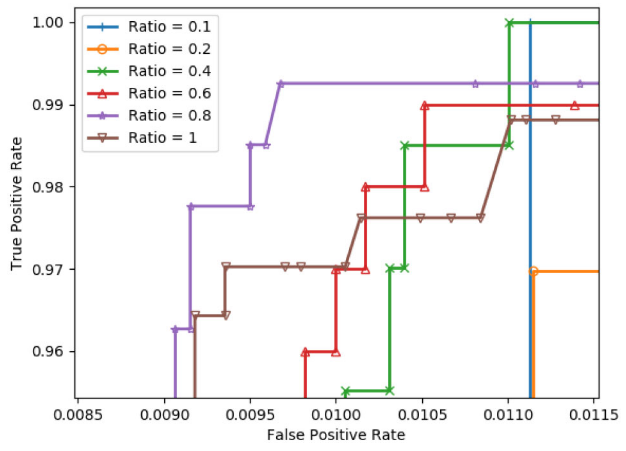


Fig. 10. ROC curve for decrease in number of TAs with different *Ratio*, zoomed on top-left corner.

TABLE XIV
DECREASE IN NUMBER OF TAS WITH DIFFERENT *Ratio*

Ratio	Precision	Recall	F1 score	TP	FP	FN
1	60.58%	98.81%	0.751	166	108	2
0.8	53.85%	99.25%	0.6982	133	114	1
0.6	45%	99%	0.6188	99	121	1
0.4	34.54%	100%	0.513	67	127	0
0.2	19.76%	100%	0.3299	33	134	0
0.1	10.46%	100%	0.189	16	137	0

Decreasing the *Ratio* can decrease the value of features related to sparseness (*sparseFeatures*). In fact, when attacking fewer TAs, the *ODS* of the attacking (source) host will decrease. Consequentially, the $WIDS_{(usr,src)}(dst_j)$ feature will also decrease, as it is weighted by the source *ODS*. We assert this claim by inspecting the features for each host in our experiments. The decrease in *Ratio* does not severely impact the true positive rate (TPR) and the false positive rate (FPR). However, upon zooming on the upper left corner (i.e., the balance between TPR and FPR), as shown in Fig. 10, there is a noticeable variation in TPR at the balance point. We further highlight the corresponding results in Table XIV, where the number of FP increases as the *Ratio* decreases.

By inspecting the data points after sampling, we observe that for most non-TAs the *sparseFeatures* remain at a very low level. For example, the average $WIDS_{(usr,src)}(dst_j)$ of TAs is around ten times that of non-TAs. Furthermore, we notice a drop in these features for TAs. Therefore, we concur that the classifier is not aggressively relying on these distinctive features. One way to adjust the sensitivity to *sparseFeatures* is to tune the classifier threshold. Table XV shows the result with the threshold values that lead to the best F1 score. There is a clear increase in precision when compared to the use of a uniform threshold. However, manually tuning the classifier threshold is impractical after deployment. Therefore, to stress on *sparseFeatures*, we further explore by only considering these particular features.

Table XVI shows the result using *sparseFeatures* only, i.e., excluding features, such as $ID_{src}(dst_j)$, $ID_{usr}(dst_j)$, and $ID_{(usr,src)}(dst_j)$. Evidently, the classifier does not perform well at the decision boundary where TAs and non-TAs are very similar. Therefore, there is a significant increase in FP and FN, in comparison to Table XIV. With these observations, it is clear that *sparseFeatures* can effectively identify TAs, but at the same time, mis-classify non-TAs that have similar probing behavior as TAs. Furthermore, non-*sparseFeatures* serve as a support to rule out these FNs, thus it is not possible to ensure classifier robustness to smaller *Ratio* perturbations by leveraging only the *sparseFeatures*.

TABLE XV
DECREASE IN NUMBER OF TAS WITH DIFFERENT *Ratio*, AND CLASSIFIER THRESHOLD ACCORDING TO F1 SCORE

Ratio	Precision	Recall	F1 score	TP	FP	FN	Threshold
1	60.58%	98.81%	0.751	166	108	2	0.3
0.8	54.29%	99.25%	0.7024	133	112	1	0.32
0.6	45.37%	98%	0.620	98	118	2	0.33
0.4	34.74%	98.51%	0.5136	66	124	1	0.33
0.2	20.75%	100%	0.3438	33	126	0	0.35
0.1	11.03%	100%	0.1988	16	129	0	0.36

TABLE XVI
DECREASE IN NUMBER OF TAS WITH DIFFERENT *Ratio*, USING ONLY *sparseFeatures*

Ratio	Precision	Recall	F1 score	TP	FP	FN
1	54.05%	95.24%	0.690	160	136	8
0.8	47.41%	95.52%	0.634	128	142	6
0.6	39.34%	96%	0.558	96	148	4
0.4	30.14%	98.51%	0.4615	66	153	1
0.2	17.10%	100%	0.292	33	160	0
0.1	8.38%	93.75%	0.1538	15	164	1

TABLE XVII
INCREASE IN NUMBER OF TAS WITH DIFFERENT *Ratio*, USING UNIFORM THRESHOLD

Ratio	Precision	Recall	F1 score	TP	FP	FN
1	60.58%	98.81%	0.751	166	108	2
2	74.76%	94.35%	0.834	317	107	19
4	86.43%	92.86%	0.895	624	98	48
6	90.68%	91.67%	0.912	924	95	84
8	93.37%	92.26%	0.928	1240	88	104
10	94.38%	91.96%	0.932	1545	92	135

We then investigate when attackers increase the number of TAs. There is no significant drop in TPR and FPR, asserting classifier robustness to this kind of perturbation. Table XVII shows a significant increase in precision, while there is a decrease in recall. Since we choose more non-TAs as TAs and perform similar probing activities on them, the classifier is able to identify them with higher precision. However, it is counter intuitive for FN to increase significantly, since the classifier has seen similar activities. Therefore, we further investigate with only *sparseFeatures*. In comparison to Table XVII, Table XVIII shows more stable results with respect to recall. Therefore, using only *sparseFeatures* achieves stable recall at the cost of lower precision.

2) Perturbing Attack Frequency: Another aspect that we explore is the frequency of an attacker's probing activity (PA). Here we define *Ratio* as follows:

$$Ratio = \frac{\text{Number of PAs after sampling}}{\text{Number of PAs in the original dataset}}$$

TABLE XVIII
INCREASE IN NUMBER OF TAS WITH DIFFERENT *Ratio*, USING ONLY *sparseFeatures*

Ratio	Precision	Recall	F1 score	TP	FP	FN
1	54.05%	95.24%	0.690	160	136	8
2	70.55%	95.54%	0.812	321	134	15
4	83.99%	95.24%	0.893	640	122	32
6	88.96%	95.93%	0.923	967	120	41
8	91.86%	95.68%	0.937	1286	114	58
10	93.44%	95.83%	0.946	1610	113	70

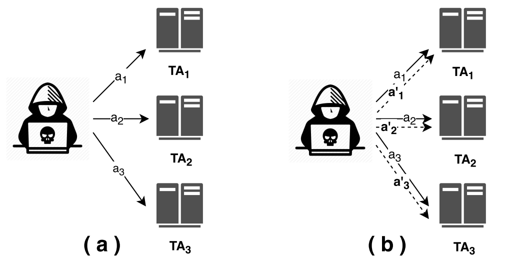


Fig. 11. Example of increase in attack frequency with $Ratio = 2$: (a) the original attack, (b) for each attack a_i on the original TA_i, we add randomness to its attack time, which results in a_i' that is simulated on TA_i.

TABLE XIX
DECREASE IN MALICIOUS AUTHENTICATION EVENTS FREQUENCY WITH DIFFERENT *Ratio*

Ratio	Precision	Recall	F1 score	TP	FP	FN
1	60.58%	98.81%	0.751	166	108	2
0.8	60.73%	99.40%	0.754	167	108	1
0.6	60.73%	99.40%	0.754	167	108	1
0.4	60.73%	99.40%	0.754	167	108	1
0.2	60.87%	100%	0.757	168	108	0
0.1	60.87%	100%	0.757	168	108	0

For increasing the attack frequency, we simulate additional attacks by replicating the original attacks on a TA by a factor of *Ratio*, as shown in Fig. 11. For each new attack we also add randomness to its attack time. Similarly, for a decrease in attack frequency, we randomly sample the attacks on each TA to the *Ratio* of the original attack.

Since the features are built atop sparse authentication events, reducing the frequency further emphasizes the *sparseFeatures*. Thus, there is no significant drop in TPR or FPR. Fig. 12 shows the upper left corner of the ROC curve, where the optimum balance of TPR and FPR resides. In general, the model performs better as the *Ratio* decreases. Table XIX shows the precision, recall, and F1 score with decreasing *Ratio*. As the *Ratio* decreases, all metrics increase steadily. That is, as attacks get less frequent, *sparseFeatures* become more pronounced, which helps the classifier better identify TAs. Hence, our model shows robustness to infrequent attacks. However, similar to Table XVI, Table XX shows the result using *sparseFeatures* only, with the classifier underperforming (i.e., significant increase in FP and FN, in comparison to Table XIX) when TAs are very similar to non-TAs.

Fig. 13 shows the ROC curve of increasing the attack frequency. As the *Ratio* increases, probing behavior becomes similar to benign activities, which diminishes the discriminating nature of *sparseFeatures*. As a result, both the precision

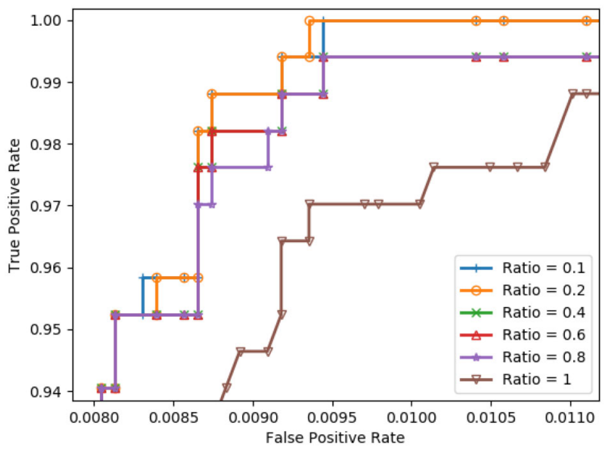


Fig. 12. ROC curve for decrease in attack frequency with different *Ratio*, zoomed on top-left corner.

TABLE XX
DECREASE IN MALICIOUS AUTHENTICATION EVENTS FREQUENCY WITH DIFFERENT *Ratio*, USING ONLY *sparseFeatures*

Ratio	Precision	Recall	F1 score	TP	FP	FN
1	54.05%	95.24%	0.690	160	136	8
0.8	54%	96.43%	0.692	162	138	6
0.6	53.85%	95.83%	0.690	161	138	7
0.4	54%	96.43%	0.692	162	138	6
0.2	53.97%	97.02%	0.694	163	139	5
0.1	54.13%	97.62%	0.696	164	139	4

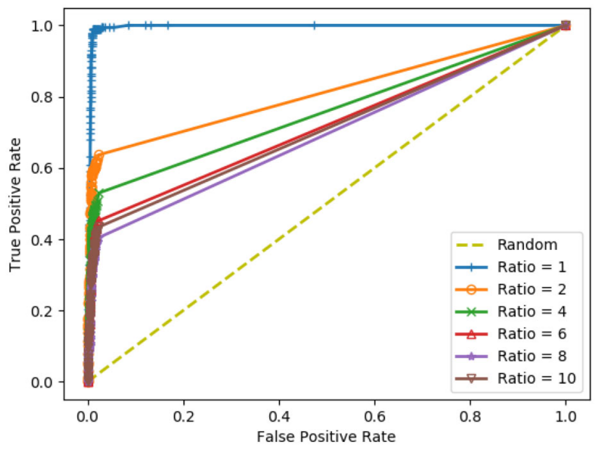


Fig. 13. ROC curve for increase in attack frequency with different *Ratio*.

and recall suffer, as shown in Tables XXI and XXII. Even though employing all features show consistently better result, the less prominent sparseness-related features sabotage model performance.

3) Adding Synthetic TAs: With the above discussion, we conclude that our model is robust when the attacker decreases the attack frequency or increases the number of TAs. But when the attacker changes the attacking pattern in the other direction (i.e., increases the attack frequency or decreases the number of TAs), our model is vulnerable (i.e., precision and recall drop significantly).

To alleviate this impact, we incorporate synthetic TAs (i.e., in the training dataset) with perturbed attack patterns to

TABLE XXI
INCREASE IN MALICIOUS AUTHENTICATION EVENTS FREQUENCY WITH DIFFERENT *Ratio*

Ratio	Precision	Recall	F1 score	TP	FP	FN
1	60.58%	98.81%	0.751	166	108	2
2	47.83%	58.93%	0.528	99	108	69
4	42.17%	41.67%	0.419	70	96	98
6	36.43%	30.36%	0.331	51	89	117
8	33.33%	26.19%	0.293	44	88	124
10	21.74%	14.89%	0.177	25	90	143

TABLE XXII
INCREASE IN MALICIOUS AUTHENTICATION EVENTS FREQUENCY WITH DIFFERENT *Ratio*, USING ONLY *sparseFeatures*

Ratio	Precision	Recall	F1 score	TP	FP	FN
1	54.05%	95.24%	0.690	160	136	8
2	40.52%	55.95%	0.47	94	138	74
4	34.33%	41.07%	0.374	69	132	99
6	27.87%	30.36%	0.291	51	132	117
8	25.99%	27.38%	0.267	46	131	122
10	16.13%	14.88%	0.155	25	130	143

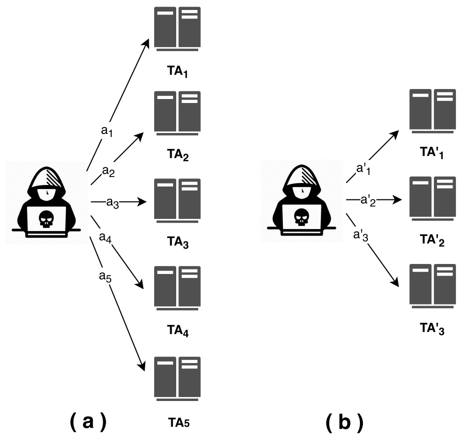


Fig. 14. Example of generating synthetic TAs from a subset of original TAs with $\gamma = 0.6$: (a) the original attack, (b) randomly choose TA'_i among the original TA_i. Add randomness to each attack a_i's attack time, which results in a'_i that is simulated on TA'_i.

enhance the classifier's ability in differentiating TAs from non-TAs in an adversarial setting. More specifically, we replace the original TAs with synthetic ones that we generate by randomly choosing a subset of the original TAs, with ratio γ, and adding randomness to their attack time (i.e., for corresponding authentication events), as shown in Fig. 14. We retrain our model and reevaluate its performance. After experimenting, we get the best result with $\gamma = 0.6$.

Table XXIII depicts the classifier performance after adding these synthetic TAs to the training dataset. Comparing the results with that of the original experiment (see Table XVI), there are more FPs, which means lower precision, while the recall is restored to 100%. Adding the synthetic TAs in the training dataset makes the classifier more aggressive at the boundary. This results in a higher number of FPs, while enabling a 100% recall, our primary objective. Table XXIV shows a positive influence on performance when the adversary increases the attack frequency (when compared to Table XXI). Interestingly, replacing the original TAs with a

TABLE XXIII
DECREASE IN NUMBER OF TAs AFTER REPLACING WITH SYNTHETIC TAs IN TRAINING DATASET ($\gamma = 0.6$)

Ratio	Precision	Recall	F1 score	TP	FP	FN
1	53.53%	99.40%	0.6958	167	145	1
0.8	46.85%	100%	0.638	134	152	0
0.6	38.61%	100%	0.557	100	159	0
0.4	28.88%	100%	0.448	67	165	0
0.2	16.10%	100%	0.277	33	172	0
0.1	8.38%	100%	0.155	16	175	0

TABLE XXIV
INCREASE IN MALICIOUS AUTHENTICATION EVENTS FREQ AFTER REPLACING WITH SYNTHETIC TAs IN TRAINING DATASET ($\gamma = 0.6$)

Ratio	Precision	Recall	F1 score	TP	FP	FN
1	53.53%	99.40%	0.6958	167	145	1
2	46.51%	59.92%	0.522	100	115	68
4	41.08%	45.24%	0.431	76	109	92
6	33.74%	32.74%	0.332	55	108	113
8	31.41%	29.17%	0.302	49	107	119
10	21.43%	17.86%	0.195	30	110	138

TABLE XXV
DECREASE IN NUMBER OF TAs AFTER REPLACING WITH SYNTHETIC TAs IN TRAINING DATASET ($\gamma = 0.1$)

Ratio	Precision	Recall	F1 score	TP	FP	FN
1	53.55%	98.81%	0.6946	166	144	2
0.8	47.00%	99.25%	0.638	133	150	1
0.6	38.67%	99%	0.556	99	157	1
0.4	29.13%	100%	0.451	67	163	0
0.2	16.26%	100%	0.280	33	170	0
0.1	8.47%	100%	0.156	16	173	0

TABLE XXVI
INCREASE IN MALICIOUS AUTHENTICATION EVENTS FREQ AFTER REPLACING WITH SYNTHETIC TAs IN TRAINING DATASET ($\gamma = 0.1$)

Ratio	Precision	Recall	F1 score	TP	FP	FN
1	53.55%	98.81%	0.6946	166	144	2
2	46.70%	58.93%	0.521	99	113	69
4	42.13%	44.64%	0.434	75	103	93
6	33.77%	30.95%	0.323	52	102	116
8	31.54%	27.98%	0.297	47	102	121
10	22.73%	17.86%	0.2	30	102	138

lower number of synthetic TAs in the training dataset alleviates both vulnerabilities. Recall that perturbing attack behavior makes the *sparseFeatures* less prominent, causing the classifier to misclassify TAs with lower sparseness-related features. Adding synthetic TAs enable the classifier to correctly classify TAs with less prominent *sparseFeatures*, making it robust to adversarial attempts.

We expect this influence to decrease as the number of synthetic TAs added to the training dataset decreases. Tables XXV and XXVI assert our conjecture with $\gamma = 0.1$, where the recall degrades with a reduced number of synthetic TAs. Therefore, we add synthetic TAs with $\gamma = 0.6$ in the training dataset, to achieve the best recall at the cost of slightly lower precision. However, in the case when an attacker increases the attack frequency, the FN is still very high (see Table XXIV). Therefore, we also add synthetic TAs generated by simulating this with $\gamma = 0.6$. We do observe slightly better result, as

TABLE XXVII
DECREASE IN NUMBER OF TAs AFTER ADDING SYNTHETIC TAs TO TRAINING DATASET ($\gamma = 0.6$)

Ratio	Precision	Recall	F1 score	TP	FP	FN
1	54.61%	98.81%	0.703	166	138	2
0.8	47.84%	99.25%	0.646	133	145	1
0.6	39.13%	99%	0.561	99	154	1
0.4	29.65%	100%	0.457	67	159	0
0.2	15.14%	100%	0.263	33	185	0
0.1	7.92%	100%	0.147	16	186	0

TABLE XXVIII
INCREASE IN MALICIOUS AUTHENTICATION EVENTS FREQ AFTER ADDING SYNTHETIC TAs TO TRAINING DATASET ($\gamma = 0.6$)

Ratio	Precision	Recall	F1 score	TP	FP	FN
1	54.61%	98.81%	0.703	166	138	2
2	45%	58.93%	0.510	99	121	69
4	38.5%	45.83%	0.418	77	123	91
6	29.33%	36.31%	0.324	61	147	107
8	27.72%	33.33%	0.303	46	146	112
10	17.24%	20.83%	0.189	35	168	133

shown in Table XXVIII. However, Table XXVII shows slightly worse performance in comparison to Table XXIII, where the synthetic TAs are simulating the attackers increasing the attack frequency. On inspecting the features for TAs when increasing the attack frequency, we notice that the *sparseFeatures* are much less prominent, making TAs very similar to non-TAs. This is because our model inherently detects sparse probing behaviour. Therefore, when the attack frequency increases too much, the attack itself becomes similar to non-TA behaviour, which results in it being filtered out by our model. Attacks in this category can be detected by models that target nosier probing behaviour.

In the two vulnerable cases of attackers increasing the attack frequency and probing smaller number of TAs, our approach results in an effective mitigation of adversarial attempts against our model. However, we still notice high FN in Table XXIV. We have explored adding synthetic TAs that are generated by reducing γ to change the number of TAs. Further adjusting the number of synthetic TAs can potentially improve the performance of our model. Besides, adding synthetic TAs generated by increasing the frequency may also further mitigate the impact of the perturbation. However, improvement in the latter case would be marginal, since our model is not designed for noisier attack patterns.

V. CONCLUSION

We propose a novel approach for detecting TAs during the LM phase of an APT attack. We explore graph-based features extracted from host authentication logs in the LANL dataset. Among all the baseline features, we filter less impactful and correlated features to select the ideal feature set for TA detection, and reduce computational overhead. To cope with the highly imbalanced nature of the dataset, different sampling algorithms are explored to improve classifier performance. The result shows that our approach is robust against imbalanced dataset. Our approach outperforms the other state-of-the-art approaches in TA detection on the LANL dataset. Furthermore, in situations where attackers evade detection by adding variation in their probing behaviour, we show that our model can be robust by adding synthetic TAs in the training data.

Due to the poor quality of flow data in the LANL dataset, we are unable to exploit data from multiple sources for TA detection. Furthermore, the employed graph-based features are generated in SF with parameters that must be further explored with different benign activity patterns. An adaptive SF that accommodates to benign behavior can also improve the robustness of TA detection. Though in our approach we focus on maximizing recall, high false alarms can result in an overhead for security experts. Therefore, we will work on further improving the F1 score of our model in the future. Lastly, as the data grows rapidly in an enterprise network, the use of online learning would be valuable, both in terms of computation overhead and performance. This will facilitate the adjustment of ML decision boundary after deployment.

REFERENCES

[1] S. Gatlan. (Mar. 2019). *Asus Live Update Infected With Backdoor in Supply Chain Attack*. Accessed: Apr. 5, 2019. [Online]. Available: https://www.bleepingcomputer.com/news/security/asus-live-update-infected-with-backdoor-in-supply-chain-attack/

[2] *Crypto Exchange Dragonex Lost $7m in Hack, Announces Compensation Plan*, TokenPost, Apr. 2019. Accessed: Apr. 5, 2019. [Online]. Available: https://tokenpost.com/Crypto-exchange-DragonEx-lost-7M-in-hack-announces-compensation-plan-1568

[3] S. Sinclair, S. W. Smith, S. Trudeau, M. E. Johnson, and A. Portera, "Information risk in financial institutions: Field study and research roadmap," in *Enterprise Applications and Services in the Finance Industry*. Heidelberg, Germany: Springer, 2008.

[4] J. H. Saltzer and M. D. Schroeder, "The protection of information in computer systems," *Proc. IEEE*, vol. 63, no. 9, pp. 1278–1308, Sep. 1975.

[5] S. Ayoubi *et al.*, "Machine learning for cognitive network management," *IEEE Commun. Mag.*, vol. 56, no. 1, pp. 158–165, Jan. 2018.

[6] R. Boutaba *et al.*, "A comprehensive survey on machine learning for networking: Evolution, applications and research opportunities," *J. Internet Serv. Appl.*, vol. 9, no. 1, p. 16, 2018.

[7] A. A. Daya, M. A. Salahuddin, N. Limam, and R. Boutaba, "BotChase: Graph-based bot detection using machine learning," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 1, pp. 15–29, Mar. 2020.

[8] G. Kaiafas *et al.*, "Detecting malicious authentication events trustfully," in *Proc. IEEE/IFIP Netw. Oper. Manag. Symp. (NOMS)*, Taipei, Taiwan, 2018, pp. 1–6.

[9] A. D. Kent, *Comprehensive, Multi-Source Cyber-Security Events*, Los Alamos Nat. Lab., Los Alamos, NM, USA, 2015.

[10] H. Bian, T. Bai, M. A. Salahuddin, N. Limam, A. A. Daya, and R. Boutaba, "Host in danger? detecting network intrusions from authentication logs," in *Proc. 15th Int. Conf. Netw. Serv. Manag. (CNSM)*, 2019, pp. 1–9.

[11] P. Chen, L. Desmet, and C. Huygens, "A study on advanced persistent threats," in *Proc. IFIP Int. Conf. Commun. Multimedia Security*, 2014, pp. 63–72.

[12] *Detecting Lateral Movement Through Tracking Event Logs*, JPCERT/CC, Tokyo, Japan, May 2017. [Online]. Available: https://www.jpcert.or.jp/english/pub/sr/ir_research.html

[13] (2018). *The MITRE Corporation*. [Online]. Available: https://attack.mitre.org/

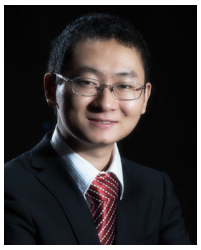
[14] P. Garcia-Teodoro, J. Díaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," *Comput. Security*, vol. 28, nos. 1–2, pp. 18–28, 2009.

[15] Q. Liu *et al.*, "Latte: Large-scale lateral movement detection," in *Proc. IEEE Mil. Commun. Conf.*, Los Angeles, CA, USA, 2018, pp. 1–6.

[16] M. Chen, Y. Yao, J. Liu, B. Jiang, L. Su, and Z. Lu, "A novel approach for identifying lateral movement attacks based on network embedding," in *Proc. IEEE Int. Conf. Parallel Distrib. Process. Appl. Ubiquitous Comput. Commun. Big Data Cloud Comput. Soc. Comput. Netw. Sustain. Comput. Commun. (ISPA/IUCC/BDCloud/SocialCom/SustainCom)*, 2018, pp. 708–715.

[17] A. Bohara, M. A. Noureddine, A. Fawaz, and W. H. Sanders, "An unsupervised multi-detector approach for identifying malicious lateral movement," in *Proc. IEEE 36th Symp. Rel. Distrib. Syst. (SRDS)*, Hong Kong, China, 2017, pp. 224–233.
[18] P. Van Mieghem, "The N-intertwined SIS epidemic network model," *Computing*, vol. 93, pp. 147–169, Oct. 2011.
[19] A. Chawla, B. Lee, S. Fallon, and P. Jacob, "Host based intrusion detection system with combined CNN/RNN model," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discov. Databases Workshops*, 2019, pp. 149–158.
[20] G. Creech and J. Hu, "Generation of a new IDS test dataset: Time to retire the KDD collection," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Shanghai, China, 2013, pp. 4487–4492.
[21] A. R. Tuor, R. Baerwolf, N. Knowles, B. Hutchinson, N. Nichols, and R. Jasper, "Recurrent neural network language models for open vocabulary event-level cyber anomaly detection," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 285–293.
[22] A. Brown, A. Tuor, B. Hutchinson, and N. Nichols, "Recurrent neural network attention mechanisms for interpretable system log anomaly detection," in *Proc. Workshop Mach. Learn. Comput. Syst.*, 2018, pp. 1–8.
[23] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Proc. EEE Eur. Symp. Security Privacy (EuroS&P)*, 2016, pp. 372–387.
[24] N. Papernot, P. McDaniel, A. Sinha, and M. Wellman, "Towards the science of security and privacy in machine learning," 2016. [Online]. Available: arXiv:1611.03814.
[25] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Trans. Evol. Comput.*, vol. 23, no. 5, pp. 828–841, Oct. 2019.
[26] W. Xu, Y. Qi, and D. Evans "Automatically evading classifiers," in *Proc. 23rd Annu. Netw. Distrib. Syst. Security Symp. (NDSS)*, 2016, pp. 21–24.
[27] B. Biggio *et al.*, "Evasion attacks against machine learning at test time," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discov. Databases*, 2013, pp. 387–402.
[28] H. S. Anderson, J. Woodbridge, and B. Filar, "DeepDGA: Adversarially-tuned domain generation and detection," in *Proc. ACM Workshop AI Security*, 2016, pp. 13–21.
[29] P. Baldi, "Autoencoders, unsupervised learning and deep architectures," in *Proc. Int. Conf. Unsupervised Transfer Learn.*, 2011, pp. 37–50.
[30] P. Hart, "The condensed nearest neighbor rule (corresp.)," *IEEE Trans. Inf. Theory*, vol. IT-14, no. 3, pp. 515–516, May 1968.
[31] D. L. Wilson, "Asymptotic of nearest neighbor rules using edited data," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-2, no. 3, pp. 408–421, Jul. 1972.
[32] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. AI Res.*, vol. 16, pp. 321–357, Jun. 2002.
[33] H. He, Y. Bai, E. A. Garcia, and S. Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," in *Proc. IEEE Int. Joint Conf. Neural Netw. World Congr. Comput. Intell. (IJCNN)*, Hong Kong, China, 2008, pp. 1322–1328.
[34] H. Bian, T. Bai, M. A. Salahuddin, N. Limam, A. A. Daya, and R. Boutaba. (2020). *Technical Report*. [Online]. Available: http://bit.ly/tech_report_2020
[35] T. E. Oliphant, *Guide to NumPy*, 2nd ed., North Charleston, SC, USA: CreateSpace Independent Publ. Platform, 2015.
[36] E. Jones *et al.*. (2001). *SciPy: Open Source Scientific Tools for Python*. Accessed: Mar. 2019. [Online]. Available: http://www.scipy.org/
[37] W. McKinney, "Data structures for statistical computing in python," in *Proc. Python Sci. Conf.*, 2010, pp. 51–56.
[38] G. Lemaître, F. Nogueira, C. K. Aridas, "Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning," *J. Mech. Learn. Res.*, vol. 18, no. 17, pp. 1–5, 2017.
[39] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in python," *J. Mech. Learn. Res.*, vol. 12, pp. 2825–2830, Nov. 2011.
[40] L. Tolosi and T. Lengauer, "Classification with correlated features: Unreliability of feature ranking and solutions," *Bioinformatics*, vol. 27, no. 14, pp. 1986–1994, 2011.
[41] L. I. Kuncheva, *Combining Pattern Classifiers: Methods and Algorithms*. Hoboken, NJ, USA: Wiley, 2004.
[42] T. G. Dietterich, "Ensemble methods in machine learning," in *Proc. Int. Workshop Multiple Classifier Syst. (MCS)*, 2000, pp. 1–15.

Haibo Bian received the B.S.E. degree from Zhejiang University in 2016, and the M.Math. degree in computer science from the University of Waterloo in 2019. His current research interests include the network functions virtualization, cybersecurity, and machine learning.



Tim Bai received the B.C.S. degree and the M.Math. degree in computer science from the University of Waterloo in 2017 and 2019, respectively. His current research interests include the network softwarization, cybersecurity, and machine learning.

Mohammad A. Salahuddin (Member, IEEE) received the Ph.D. degree in computer science from Western Michigan University in 2014. He is a Research Assistant Professor with the David R. Cheriton School of Computer Science, University of Waterloo. His current research interests include the Internet of Things, content delivery networks, network softwarization, cloud computing, and cognitive network management. He serves as a TPC member for international conferences and a reviewer for various journals and magazines.

Noura Limam received the M.Sc. and Ph.D. degrees in computer science from the University Pierre and Marie Curie (Sorbonne University) in 2002 and 2007, respectively. She is currently a Research Assistant Professor of Computer Science with the University of Waterloo. Her contributions are in the area of network and service management. Her current research interests are in network softwarization and cognitive network management.

Abbas Abou Daya received the B.Eng. degree in electrical and computer engineering from the American University of Beirut, and the M.Math. degree in computer science from the University of Waterloo in 2019. His current research interests involve machine learning, cybersecurity, and networks and systems.

Raouf Boutaba (Fellow, IEEE) received the M.Sc. and Ph.D. degrees in computer science from Sorbonne University in 1990 and 1994, respectively. He is currently a University Chair Professor and the Director of the David R. Cheriton School of Computer Science, University of Waterloo, Canada. He also holds an INRIA International Chair in France. He was the Founding Editor-in-Chief of the IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT from 2007 to 2010. He is currently the Editor-in-Chief of the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS. He is a fellow of the Engineering Institute of Canada, the Canadian Academy of Engineering, and the Royal Society of Canada.