Host in Danger? Detecting Network Intrusions from Authentication Logs

Haibo Bian, Tim Bai, Mohammad A. Salahuddin, Noura Limam, Abbas Abou Daya, and Raouf Boutaba
University of Waterloo, Waterloo Ontario N2L 3G1, Canada
{haibo.bian, tim.bai, mohammad.salahuddin, n2limam, aaboudaya, rboutaba}@uwaterloo.ca

***Abstract*—Recently, network infiltrations due to advanced persistent threats (APTs) have grown significantly, resulting in considerable losses to businesses and organizations. APTs are stealthy attacks with the primary objective of gaining unauthorized access to network assets. They often remain dormant for an extended period of time, which makes their detection challenging. In this paper, we leverage machine learning (ML) to detect hosts in a network that are targeted by an APT attack. We evaluate a number of ML classifiers to detect susceptible hosts in the Los Alamos National Lab dataset. We explore (i) graph-based features extracted from multiple data sources *i.e.*, network flows and host authentication logs, (ii) feature engineering to reduce dimensionality, and (iii) balancing the training dataset using numerous over- and under-sampling techniques. Finally, we compare our 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.**

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

I. Introduction

Cyber attacks have been growing in sophistication, resulting in considerable damage to businesses. They not only result in financial losses, but also impact customer trust and churn. There has been an increasing trend in cyber attacks in recent years. 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 a 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 attacker has gained persistent access to certain network resources (*e.g.*, servers or end-hosts). The goal of LM is to infiltrate other resources and gain higher privileges inside the target network. This is typically achieved by performing credential stealing or vulnerability exploitation on already compromised hosts. Interestingly, 50%–90% of employees have access to data that they no longer need [3]. This is primarily due to poor security practises, such as the violation of the least privilege principle [4], which increases the likelihood of an attacker penetrating the crucial network assets via LM. Therefore, it is vital to detect LM at an early stage.

As opposed to the traditional detection of successful intrusions [5], an alternative 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, which 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.

Early detection of LM can be addressed by (i) tagging the malicious host events, or (ii) tagging the target assets (TA). However, the stealthiness and sparseness of malicious events can make the first strategy a difficult endeavor. Though, crafting discriminative features for each event can achieve high recall, it comes at a high computational overhead. This makes the first strategy unscalable for very large networks. Furthermore, for complex network infrastructure with sporadic events, tagging individual events can also result in a high number of false positives. In comparison, tagging TAs reduce computational overhead. With carefully crafted features from sparse events, it is possible to achieve a high precision in detection performance (*cf.*, Section IV). We focus on the second strategy for early detection of LM by leveraging host authentication logs.

Anomaly-based methods are widely used for intrusion detection. These methods first establish a baseline of normal system behavior and model a decision engine. The decision engine determines and alerts any divergence or statistical deviations from the norm as a threat. Machine learning (ML) [6], [7] is an ideal technique to automatically establish the normal behavior of a system. However, an important step prior to training a ML model is feature extraction. These features act as discriminators for learning and inference, and increase the accuracy of ML models. The most commonly employed features in 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

978-3-903176-24-9 © 2019 IFIP

capture the host communication patterns that may expose additional aspects of malicious behavior. Graph-based features, derived from flow-level or event-level information to reflect the true behaviour of hosts, are an alternative that overcome this limitation.

The distribution of the dataset can also severely influence ML performance. For example, in the case of an imbalanced dataset (*e.g.*, sparse malicious host events versus benign events), the ML techniques are more likely to classify new data to the majority class. Though, balancing the dataset can alleviate this issue, it may sabotage ML performance by impacting graph-based features (*cf.*, Section IV). In this paper, we propose a novel approach for anomaly-based early detection of LM. Our main contributions are:

- We leverage ML for identifying TAs to facilitate early detection of LM. In this respect, we evaluate numerous supervised ML techniques and their ensemble, and compare them in classification performance and overhead.
- We employ graph-based features using real datasets from the Los Alamos National Lab (LANL) [8]. We explore features that are extracted from multiple data sources *i.e.*, network flows as well as host authentication logs, and employ feature engineering to reduce dimensionality.
- Due to the highly imbalanced nature of the LANL dataset, we evaluate various over- and under-sampling techniques, and explore their impact on ML performance.
- We 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.

The rest of the paper is organized as follows. Section II highlights the recent related works on LM detection. In Section III, we discuss the characteristics of the LANL dataset, delineate the explored sampling algorithms, expose feature extraction and selection, and present the ML techniques and evaluation metrics employed for TA detection. The results of our evaluation and comparison to the state-of-the-art approaches for LM detection are discussed in Section IV. We conclude in Section V with an outline of future research directions.

II. Related Works

ML has been extensively used for LM detection. Chen *et al.* [9] 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 neighbors (k-NN) and achieve an average of 91.3% precision in LM detection. However, their evaluation is limited to k-NN.

Bohara *et al.* [10] propose an unsupervised approach to detect malicious LM. They employ the LANL dataset and inject artificial attacks into the original dataset, instead of using redteam events. However, these simulated attacks may not depict behavior of real attacks in enterprise networks. Their LM activity simulation follows the susceptible-infected-susceptible virus spread model [11]. 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 one employs PCA and extreme value analysis. This combination achieves an 88.7% true positive rate.

Tuor *et al.* [12] and Brown *et al.* [13] propose recurrent neural network (RNN) for log level anomaly detection. Tuor *et al.* introduce a language modeling framework for generic log anomaly 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 models directly leverage tokenized log lines. They achieve an area under the receiver operating characteristics (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 at the log level.

Several other works [14]–[16] propose hybrid IDSs. Kim *et al.* [14] propose a hierarchical approach that decomposes normal training data into smaller subsets using decision tree (DT) and leverage one-class support vector machine (SVM) for each subset. Chitrakar *et al.* [15] propose a similar approach, where the training data is split into different clusters using k-medoids, followed by naïve bayes for further attack classification. Agarwal *et al.* [16] normalize entropy of network features using a custom algorithm and leverage SVM for attack classification. All of the aforementioned approaches combine multiple techniques for classification, but none of them leverage data from different sources. In contrast, we explore features extracted from multiple data sources to improve classification performance.

Our work is inspired by Kaiafas *et al.* [17]. They construct a bipartite graph to extract graph-based features and employ an ensemble of ML models to improve classification performance. 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.

III. Methodology

A. Dataset

1) Characteristics: The LANL dataset contain logs from multiple data sources, including authentication log, flow log, DNS log, and process log. We explore the authentication and flow logs for TA detection during LM.

a) Authentication Log: This log is composed of over 450 million authentication events from Windows-based desktop computers, spanning 58 days. Among these events there are 749 redteam compromise events, distributed in the first 30 days of the dataset, as depicted in Fig. 1. We leverage data in this time frame, which consists of about 230 million events from 14,582 benign hosts and 299 redteam related hosts. However, the malicious activities are a very small fraction of all the activities in the LANL dataset.

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 scope for this work. 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*, can potentially facilitate the detection of TAs.

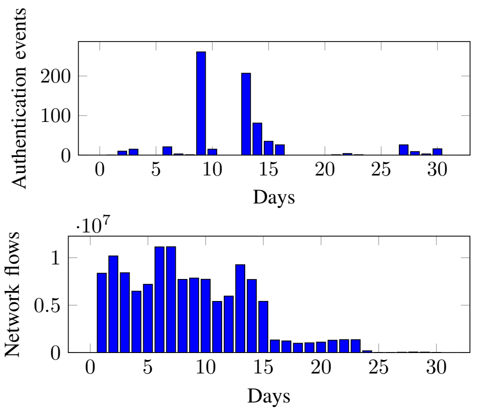

Figure 1. Redteam activities and network flow distributions

b) Flow Log: This log contains flow events collected from the central routers in the network. It spans 30 days with a total of 129 million flows, as shown in Fig. 1. There is a noticeable change in distribution from day 16, which indicates a major change in the system. It is also worth mentioning that flows that do not go through the central routers will not be recorded. As a result, out of the 14,881 hosts in the authentication log, only 3,456 hosts have corresponding flow data. Also, due to a misconfiguration of internal network routers, the flow data collection completely stops after day 29 [18].

2) Balancing: Sampling algorithms are employed when a dataset is highly imbalanced. An imbalanced dataset can result in a classifier that is biased on the majority class, due to the nature of the training procedure. The sampling algorithms 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 [19]. However, a potential advantage of under-sampling is the reduced computational overhead. On the 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 [20]. This algorithm keeps all samples in minority class and uses 1-NN classifier to determine whether to retain the data point in majority class or not. The next algorithm is Repeated Edited Nearest Neighbours (RENN), which implements multiple iterations of Edited Nearest Neighbours (ENN) [21]. For the over-sampling algorithms, we start with random over-sampling (ROS), followed by the well-known synthetic minority over-sampling technique (SMOTE) [22]. 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) [23]. 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 purity of graph-based features. For example, all authentication events pertaining to a 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 35 features: 29 features from the authentication log and 6 from the flow log. A detailed description of the extracted flow log-based features can be found in our previous work [24].

The 29 authentication-based features are extracted from a graph representation of the authentication events. As the features are primarily based on the in-degree 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 represent 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 u as source and v as destination. Authentication events are inserted in the graph as shown in Fig. 2.

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 logging into $ComPtr4017$ from $ComPtr10099$ twice on Day_1, 3 times on Day_n, but never before on Day_2, the event e is added to the edge linking $ComPtr10099$ to $ComPtr4017$ on the graph G with a count of 1, as depicted in Fig. 2. Once the graph G is complete, we build dictionaries that are used to extract the features as described in [25][1].

[1]Due to space limitation, we provide a technical report to detail our feature extraction process. This will also facilitate the reproducibility of results.

Table I
MOST SIGNIFICANT FEATURES EXTRACTED FROM AUTHENTICATION LOGS

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 defined 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
$AS(Host_j)$	$MSF(Host_j) * SUR(Host_j)$

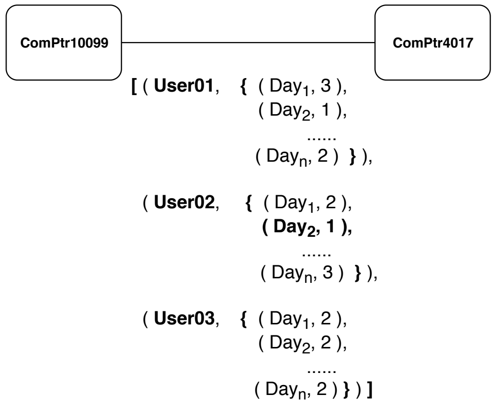

Figure 2. Graph representation of authentication events

A high-level description of the authentication-based features is provided below. Table I further delineates a subset of the authentication-based features.

a) In-Degree (ID) and Out-Degree (OD): In the early phase of LM, attackers use stolen credential 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 would 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* occur, and assigns a higher score to more infrequent events. This amplifies the impact of such events on graph 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 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 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 likely 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 in precision and recall. It serves as an evident sign that a host has been tempted by an actively probing source host. It further reveals that selected ML models cannot capture the product relation of different features. With all the features, the classifier can better distinguish the boundary for TAs and benign hosts.

Further details on each and every extracted feature is available in [25].

Algorithm 1 Sparseness function (*SF*)

```
input  : Source host $Src$, username $Usr$, destination host $Dst$,
         thresholds $\theta$, $\beta$
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 numbers of days where the events occur */
3: $TotalDays \leftarrow$
4:         $Events.filter(Src, Usr, Dst).countByDays()$
5: if $TotalDays \leq \theta$ then
6:     $Sparseness \leftarrow$
7:             $max(TotalDays * \beta - Events.count(), 0)$
8: end if
9: return $Sparseness$
```

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 these ML techniques as well as other well known techniques, such as SVM, *k*-NN, and gaussian naïve bayes. However, we do not discuss them as they under perform in our evaluation.

D. Evaluation Metrics

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

$$Precision = \frac{\text{True Positive}}{\text{True Positive + False Positive}} \times 100$$

$$Recall = \frac{\text{True Positive}}{\text{True Positive + False Negative}} \times 100$$

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

The precision and recall are better criterion 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 the area under the ROC curve (AUC) to quantify ML performance.

IV. Experiments

A. Environment

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

2) Software: We leverage Numpy [26], Scipy [27], and Pandas [28] for data pre-processing. Imbalanced-learn [29] is employed for balancing the training datasets, while Scikit-learn [30] is used for building ML models.

B. Results

1) Feature Selection: We start with evaluating the performance of different ML classifiers with graph-based features extracted from authentication logs and network flow logs. Table II showcases the result of k-fold cross-validation ($k = 10$) using a total of 35 features (6 flow-based and 29 authentication-based features) extracted from the first 30 days of 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 *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 as the number of estimator with a maximum depth of 12. LB uses 100 estimators and a DT regressor with a maximum depth of 3. LR is using tolerance of 0.0001 and a regularization strength of 1.

Table II
ML PERFORMANCE USING FLOW- AND AUTHENTICATION-BASED FEATURES (35 FEATURES)

ML model	Precision	Recall	F1 score	Training time (s)
DT	75.62%	75.15%	0.75	0.14
RF	79.99%	79.27%	0.79	3.31
LB	80.31%	80.29%	0.80	6.69
LR	31.10%	5.47%	0.09	4.04

With the exception of LR, which performs poorly, the other ML techniques classify TAs with relatively high precision and recall (over 75%). LB outperforms DT and RF with the highest F1 score. The number of ML features not only influence the computational overhead, but can also result in model over-fitting. Hence, in order to reduce the number of features and identify the ideal feature set for TA detection, we then restrict the feature set to authentication-based features. As depicted in Table III, with the exception of LR whose performance increases significantly, we witness a marginal performance degradation when discarding flow-based features. RF outperforms other classifiers, while saving about $14s$ in feature extraction time. The lackluster performance of the flow-based features can be attributed to the inferior quality of flow data in the LANL dataset, as discussed in Section III. This undermines the suitability of flow-based features to detect TAs during LM in this particular dataset.

Table III
ML PERFORMANCE USING 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

Next, we study the correlation between authentication-based features to further reduce the features for TA detection. 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. The technical report [25] provides the Feature IDs (FIDs) for the authentication-based features.

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

Based on these findings, 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. This latter feature is only significant if there is evidence that the TA is compromised and is actively attempting to move laterally, which is not the case in this dataset. We re-evaluate the ML techniques after removing the 12 aforementioned features.

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 LB are immune to highly correlated features [31], 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, TAs do not necessarily have an exploring behaviour, thus 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.

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

2) Ensemble Learning: In an effort to improve the performance of the stand-alone ML models for TA detection, we consolidate them using ensemble learning. Due to the lackluster performance of LR in comparison to other ML models (*cf.*, Table V), we remove it from the list of potential classifiers in the ensemble approach. First, we employ the majority voting (MV) algorithm [32] 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. However, this results in an inferior performance to stand-alone RF, since low performing classifiers can influence the voting process.

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

Evidently, MV is unable to boost the performance of the best stand-alone classifier. Therefore, we explore another ensemble approach, namely weighted voting (WV) [33], 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 combinations of weights assigned to the ML models in Table VII, stand-alone RF still outperforms WV. Besides, these ensemble approaches increase training time, undermining their suitability for early LM detection. Therefore, we choose the stand-alone RF classifier as *our model* for further experiments.

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

RF	LB	DT	Precision	Recall	F1	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

3) Balancing the Dataset: We further 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 our model. We evaluate several well known sampling algorithms (*cf.*, 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 Benign (TA/Bening). 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 do not capture the true nature of original TAs, resulting in a lower recall. 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 benign hosts. As opposed to over-sampling only a portion of the minority points, ROS simply replicates TAs, which stresses on all TAs. Uniformly Stressing on all TAs preserve the originality of TA class and 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.

Table VIII
OVER-SAMPLING WITH DIFFERENT ALGORITHMS (17 FEATURES)

Algorithm	TA/Benign	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

b) Under-sampling: Recall that RUS randomly removes samples from the majority class. This may result in a high number of benign (majority class) hosts 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 the benign hosts are still classified with high recall. A similar affect can be seen with RENN, which removes benign hosts that are not very similar to their neighbors. In contrast, ConNN preserves the benign hosts that are different from their neighbors. Therefore, under-sampling with ConNN results in the best F1 score with precision and recall of 62.47% and 95.12%, 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 benign hosts to the largest extent, which positively impacts the training time.

Table IX
UNDER-SAMPLING WITH DIFFERENT ALGORITHMS (17 FEATURES)

Algorithm	TA/Benign	Precision	Recall	F1	ST (s)	TT (s)
RUS	0.02	60.9%	96.55%	0.7469	0.01	2.45
ConNN	0.51	62.47%	95.12%	0.7541	130.88	0.99
RENN	0.01	60.07%	97.62%	0.7437	2.81	3.32

c) Comparison: We highlight the over- and under-sampling algorithms with the highest F1 score in Table X, along with no sampling (*i.e.*, unbalanced 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 in comparison to training without any sampling. Thus, we proceed without any sampling to detect TAs during LM in the LANL dataset.

Table X
COMPARING DIFFERENT ALGORITHMS (17 FEATURES)

Algorithm	TA/Benign	Precision	Recall	F1	ST (s)	TT (s)
ADASYN	0.02	62.08%	96.07%	0.7542	0.06	4.66
ConNN	0.51	62.47%	95.12%	0.7541	130.88	0.99
Unbalanced	0.01	61.54%	95.23%	0.7476	0	3.61

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.* [9] and Kaiafas *et al.* [17]. To achieve a fair comparison, we balance the dataset according to the algorithm in [9], which preserves the redteam events while under-sampling the benign activities. Due to scalability issues in [17], 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 precision, with an improvement of 0.02 in F1 score. However, their feature engineering 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.

Table XI
TA DETECTION USING STAND-ALONE RF AND CROSS-VALIDATION VERSUS ([9], [17])

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

Followed by 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 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. The authors in [9] leverage features, including network traffic amount, sending packet amount, authentication amount and DNS queries amount, *etc*. These generic statistical features 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% and 0.75, respectively.

Table XII
ROBUSTNESS OF TA DETECTION USING STAND-ALONE RF VERSUS ([9], [17])

Classifier	Precision	Recall	F1	FET (s)	TT (s)
Our Model	60.58%	98.81%	0.75	2210.45	3.95
Chen *et al.*	3.16%	2.98%	0.030	823.51	59.45
Kaiafas *et al.*	—	—	—	$>$ 360000	—

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 disclose 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 *et al.* perform quite well in cross-validation, they fail in robustness to never seen TAs. The authors in [17] extract features, including frequency, first occurrence tag, diversity of user, *etc*. However, these features fail to differentiate the TAs from the benign hosts for large networks. Due to the diversity of different authentication events, they can result in hosts 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 filter out noise and differentiate TAs. Fig. 3 shows the ROC curve, which indicates that our model has the highest AUC score of 0.995. Note that Kaiafas's model is using MV, which is not feasible to be plotted in ROC curve. In comparison to previous robustness result, our model shows a marginal loss in precision and recall. However, our model out classes other approaches, with a high recall of over 94%, while the F1 score is the highest at 0.74.

Table XIII
ROBUSTNESS OF TA DETECTION USING STAND-ALONE RF VERSUS ([9], [17]) ON A REDUCED TRAINING DATASET

Classifier	Precision	Recall	F1	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

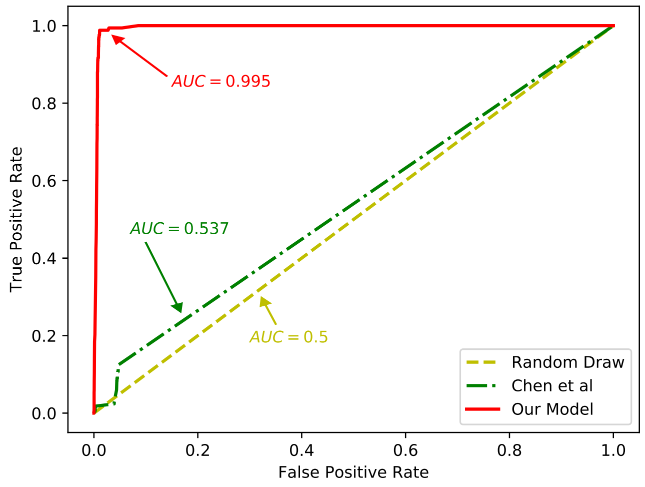

Figure 3. ROC for robustness in TA detection using stand-alone RF vs. ([9], [17]), with days 13, 14 and 15 for training, and day 9 reserved for testing

V. Conclusion

In this paper, we propose a novel approach for detecting TAs during the LM phase of an APT attack. We explore graph-based features extracted from multiple data sources (*i.e.*, network flows and 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 unbalanced dataset. We found our approach to outperform the other state-of-the-art approaches in TA detection on the LANL dataset.

Our approach is limited by the poor quality of the LANL dataset. This prevents us from exploiting data from multiple sources for TA detection. This is largely due to the incompleteness of network traffic monitoring data. Apart from this aspect, the sampling algorithms do not significantly boost the performance of classifiers, which needs further investigation. Furthermore, as the data grows rapidly in an enterprise network, the exploration of incremental learning would be valuable in the future. This will facilitate the adjustment of ML decision boundary after deployment.

Acknowledgment

This work is supported by the Royal Bank of Canada, and Partnership Grant No. 530335 of the Natural Science and Engineering Research Council of Canada.

References

[1] S. Gatlan, "Asus live update infected with backdoor in supply chain attack," Mar 2019, accessed: 2019-04-05. [Online]. Available: https://www.bleepingcomputer.com/news/security/asus-live-update-infected-with-backdoor-in-supply-chain-attack/

[2] TokenPost, "Crypto exchange dragonex lost $7m in hack, announces compensation plan," Apr 2019, accessed: 2019-04-05. [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 *Proceedings of Enterprise Applications and Services in the Finance Industry*, D. J. Veit, D. Kundisch, T. Weitzel, C. Weinhardt, F. A. Rabhi, and F. Rajola, Eds., 2008.

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

[5] S. Agrawal and J. Agrawal, "Survey on anomaly detection using data mining techniques," *Procedia Computer Science*, vol. 60, pp. 708–713, 2015.

[6] S. Ayoubi, N. Limam, M. A. Salahuddin, N. Shahriar, R. Boutaba, F. Estrada-Solano, and O. M. Caicedo, "Machine learning for cognitive network management," *IEEE Communications Magazine*, vol. 56, no. 1, pp. 158–165, 2018.

[7] R. Boutaba, M. A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano, and O. M. Caicedo, "A comprehensive survey on machine learning for networking: evolution, applications and research opportunities," *Springer Journal of Internet Services and Applications*, vol. 9, no. 1, 2018.

[8] A. D. Kent, "Comprehensive, Multi-Source Cyber-Security Events," Los Alamos National Laboratory, 2015.

[9] 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 *Proceedings of IEEE International Conf. on Parallel Distributed Processing with Applications, Ubiquitous Computing Communications, Big Data Cloud Computing, Social Computing Networking, Sustainable Computing Communications*, 2018, pp. 708–715.

[10] A. Bohara, M. A. Noureddine, A. Fawaz, and W. H. Sanders, "An unsupervised multi-detector approach for identifying malicious lateral movement," in *Proceedings of IEEE Symposium on Reliable Distributed Systems*, 2017, pp. 224–233.

[11] P. Van Mieghem, "The n-intertwined sis epidemic network model," *Computing*, vol. 93, pp. 147–169, 2011.

[12] 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 *Proceedings of Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[13] A. Brown, A. Tuor, B. Hutchinson, and N. Nichols, "Recurrent neural network attention mechanisms for interpretable system log anomaly detection," in *Proceedings of the First Workshop on Machine Learning for Computing Systems*, 2018, pp. 1–8.

[14] G. Kim, S. Lee, and S. Kim, "A novel hybrid intrusion detection method integrating anomaly detection with misuse detection," *Expert Systems with Applications*, vol. 41, no. 4, pp. 1690–1700, 2014.

[15] R. Chitrakar and C. Huang, "Anomaly based intrusion detection using hybrid learning approach of combining k-medoids clustering and naive bayes classification," in *Proceedings of IEEE International Conference on Wireless Communications, Networking and Mobile Computing*, 2012, pp. 1–5.

[16] B. Agarwal and N. Mittal, "Hybrid approach for detection of anomaly network traffic using data mining techniques," *Procedia Technology*, vol. 6, pp. 996–1003, 2012.

[17] G. Kaiafas, G. Varisteas, S. Lagraa, R. State, C. D. Nguyen, T. Ries, and M. Ourdane, "Detecting malicious authentication events trustfully," in *Proceedings of IEEE/IFIP Network Operations and Management Symposium*, 2018, pp. 1–6.

[18] A. D. Kent, "Cybersecurity Data Sources for Dynamic Network Research," in *Proceedings of Dynamic Networks in Cybersecurity*. Imperial College Press, Jun. 2015.

[19] P. Baldi, "Autoencoders, unsupervised learning and deep architectures," in *Proceedings of the International Conference on Unsupervised and Transfer Learning Workshop*, 2011, pp. 37–50.

[20] P. Hart, "The condensed nearest neighbor rule (corresp.)," *IEEE Transactions on Information Theory*, vol. 14, no. 3, pp. 515–516, 1968.

[21] D. L. Wilson, "Asymptotic properties of nearest neighbor rules using edited data," *IEEE Transactions on Systems, Man, and Cybernetics*, no. 3, pp. 408–421, 1972.

[22] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[23] H. He, Y. Bai, E. A. Garcia, and S. Li, "Adasyn: Adaptive synthetic sampling approach for imbalanced learning," in *Proceedings of IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, 2008, pp. 1322–1328.

[24] A. A. Daya, M. A. Salahuddin, N. Limam, and R. Boutaba, "A graph-based machine learning approach for bot detection," in *Proceedings of IFIP/IEEE International Symposium on Integrated Network Management*, 2019.

[25] H. Bian. Technique report for graph features. [Online]. Available: https://bitbucket.org/gentlebian/technique-report/src/master/

[26] T. E. Oliphant, *Guide to NumPy*, 2nd ed. USA: CreateSpace Independent Publishing Platform, 2015.

[27] E. Jones, T. Oliphant, P. Peterson *et al.*, "SciPy: Open source scientific tools for Python," 2001–, accessed Mar 2019. [Online]. Available: http://www.scipy.org/

[28] W. McKinney, "Data structures for statistical computing in python," in *Proceedings of the Python in Science Conference*, 2010.

[29] G. Lemaître, F. Nogueira, and C. K. Aridas, "Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning," *Journal of Machine Learning Research*, vol. 18, no. 17, pp. 1–5, 2017.

[30] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in python," *J. Mach. Learn. Res.*, vol. 12, 2011.

[31] L. Toloşi and T. Lengauer, "Classification with correlated features: unreliability of feature ranking and solutions," *Bioinformatics*, vol. 27, no. 14, pp. 1986–1994, 2011.

[32] L. I. Kuncheva, *Combining pattern classifiers: methods and algorithms*. John Wiley & Sons, 2004.

[33] T. G. Dietterich, "Ensemble methods in machine learning," in *International workshop on multiple classifier systems*. Springer, 2000, pp. 1–15.