Computer Communications 165 (2021) 9–19

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

RDP-based Lateral Movement detection using Machine Learning

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

David R. Cheriton School of Computer Science, University of Waterloo, Ontario, Canada

ARTICLE INFO

Keywords:
Cybersecurity
Advanced Persistent Threats
Lateral Movement
Adversarial learning
Machine Learning

ABSTRACT

Detecting cyber threats has been an on-going research endeavor. In this era, Advanced Persistent Threats (APTs) can incur significant costs for organizations and businesses. The ultimate goal of cybersecurity is to thwart attackers from achieving their malicious intent, whether it is credential stealing, infrastructure takeover, or program sabotage. Every cyber attack goes through several stages before its termination. Lateral Movement (LM) is one of those stages that is of particular importance. Remote Desktop Protocol (RDP) is a method used in LM to successfully authenticate to an unauthorized host that leaves footprints on both host and network logs. In this paper, we propose to detect evidence of LM using Machine Learning (ML) and Windows RDP event logs. We explore different feature sets extracted from these logs and evaluate various supervised ML techniques for classifying RDP sessions with high precision and recall. We also compare the performance of our proposed approach to a state-of-the-art approach and demonstrate that our ML model outperforms in classifying RDP sessions in Windows event logs. In addition, we show that our model is robust against certain types of adversarial attacks.

1. Introduction

Advanced Persistent Threat (APT) is one of the most prominent cyber attacks that has the potential to cause significant damage to various organizations and businesses. It is a stealthy attack in which attackers gain unauthorized access to a network for a long period of time. According to Kaspersky Lab [1], a backdoor program, called Carbanak, caused a billion dollar in cumulative losses for a financial institution. Furthermore, more than 80 million social security numbers were siphoned from Anthem, a big health insurance company, which was only detected after nine months [2].

Most secured systems maintain a strong boundary between the internet and the intranet, thus attackers choose targets that have access to hosts behind the network security functions (*e.g.*, firewalls, intrusion prevention systems, *etc.*). It is difficult for attackers to launch attacks against assets that reside in the intranet. Thus, an attacker usually leverages social engineering techniques (*e.g.*, phishing, pretexting, baiting, *etc.*) to trick network insiders into executing malicious code or surrendering credentials. This allows the attacker to gain access to the victim's computer and gradually explore for valuable information by exploiting vulnerabilities of other intranet entities. This is commonly known as Lateral Movement (LM).

During the LM phase, attackers tend to use legitimate system tools, which make the detection of APT a challenging endeavor. However, Machine Learning (ML) techniques have been widely used for APT detection [3]. ML is an ideal tool to extract knowledge from data and learn system behavior [4]. Some research utilize a single ML model, while others combine different learning techniques to form an ensemble or a hybrid model for intrusion detection. For instance, Kaiafas et al. [5] build an ensemble classifier that leverages voting mechanism, whereas Kim et al. [6] employ both Support Vector Machine (SVM) and Decision Tree (DT) to build a two-stage classification model. These techniques demonstrate significant advances in intrusion detection.

APT detection methods generally rely either on network flow data [7–10], or host system logs [11,12] to uncover evidence of APT. Network-based intrusion detection has been well explored but has several shortcomings. Firstly, there is limited information that can be extracted from network data. For privacy concerns, it is illegal to inspect network payload without user consent [3], making it non-trivial to extract meaningful information beyond packet statistics and the basic five tuple (*i.e.*, source IP, destination IP, source port, destination port, and protocol). In addition, 72% of the recent network traffic is encrypted using protocols, such as Transport Layer Security (TLS) [13]. This makes inspection of packet's payload challenging without significantly degrading system performance. Furthermore, attackers launching APT tend to be cautious and often leverage custom protocols, making it harder to detect abnormal behavior within network data.

* Corresponding author.
E-mail addresses: tim.bai@uwaterloo.ca (T. Bai), haibo.bian@uwaterloo.ca (H. Bian), mohammad.salahuddin@uwaterloo.ca (M.A. Salahuddin), aaboudaya@uwaterloo.ca (A. Abou Daya), noura.limam@uwaterloo.ca (N. Limam), rboutaba@uwaterloo.ca (R. Boutaba).

https://doi.org/10.1016/j.comcom.2020.10.013
Received 25 May 2020; Received in revised form 14 October 2020; Accepted 20 October 2020
Available online 28 October 2020

On the other hand, host-based intrusion detection can overcome the aforementioned limitations. At the end host, data is decrypted, allowing for extraction of information, including payload entropy, packet drop rate, and login failures, which can improve detection performance. Furthermore, operating systems have built-in logging functionalities, which provide abundant information. By enabling or disabling different logging levels and policies, only useful information can be logged. There are multiple stages in APT (*cf.*, Section 2) and certain stages will leave footprints allowing for the detection of intrusion in its early stage. For example, an intruder can gain access to the target host within the intranet, but this action would generate suspicious logs on the end host.

Since the ML algorithms were designed without taking security into consideration [14], both network-based and host-based intrusion detection systems are vulnerable to attacks from adversaries. Therefore, any ML-based system must be designed with defense strategies against adversarial attacks. There are numerous works that attempt to tackle this issue (*cf.*, Section 2). For example, Marco et al. [15] present a taxonomy, identifying and analyzing attacks against ML-based systems. In addition, a variety of defense techniques are proposed in their work to protect systems from different types of adversarial attacks. Biggio et al. [14] develop systematic approaches to defend against the different types of adversarial attacks.

Remote Desktop Protocol (RDP) is designed by Microsoft to provide remote display and input capabilities, while Remote Desktop Service (RDS) is a native service on Microsoft Windows platform that implements RDP. This service is frequently used by legitimate network administrators. However, it is also a primary tool used by attackers during LM [16], since discriminating between legitimate and malicious use of this tool is challenging. We surveyed nine distinct APT incidents and five of them (*i.e.*, over 50%) used RDP during the attack. Therefore, in this paper, we detect anomalous RDP sessions based on evidence from host logs with a focus on optimizing recall. The primary contributions of this work are as follows:

- We highlight the limitations of two publicly available Windows event log datasets from Los Alamos National Laboratory (LANL) [17,18]. To overcome their limitations, we combine these two datasets while preserving their realistic properties.
- We propose an ML-based approach for detecting malicious RDP sessions. We explore different feature sets and evaluate various supervised ML techniques for classifying RDP sessions in Windows event logs.
- We compare the performance of our proposed approach to a state-of-the-art method [5], and demonstrate that our ML model outperforms in the classification of RDP sessions in Windows event logs. In addition, we show that our model is robust against certain types of adversarial attacks.

The rest of the paper is organized as follows. Section 2 provides a background and presents the current state of existing host-based intrusion detection systems. Section 3 describes the characteristics and properties of the two datasets we employ in this paper. Section 4 presents the approach of crafting our synthetic dataset based on the existing dataset. Furthermore, the features extracted from this dataset are elaborated, and ML techniques and their performance evaluation metrics are discussed. In Section 5, we delineate our evaluation results in detecting anomalous RDP sessions and benchmark the robustness of our proposed model. Section 6 highlights our main contributions and provides a brief summary of this paper. In addition, this section instigates future research directions.

2. Background and motivation

2.1. Intrusion kill-chain and lateral movement

Conventional APT detection approaches assume successful intrusions and focus on individual events. However, in recent sophisticated APT, a single adversary campaign consists of multiple small, less detectable attacks. Detecting these attacks can be challenging, as a single campaign may develop over time with multiple steps, each designed to thwart a defense and take place in a different timeline.

All attacks occurring in cyberspace have patterns that can be described as a chain of events—the intrusion kill-chain [19]. At a high-level, an APT starts with *reconnaissance*, observing and identifying a target in the network. This is followed by creating a weaponized payload. *Weaponization* of payloads typically take the form of malicious emails and attachments, which are delivered to the subject of interest. *Exploitation* starts after delivery, where the malevolent code gets triggered. While malicious code execution can be stand-alone, some malwares exploit applications on the subject's machine. This can range from OS-based bugs (*e.g.*, in RDP and PsExec) to application-based faults (*e.g.*, in live processes, such as Google Chrome and Microsoft Office). The attacker then proceeds with the *installation* of a security back-door on the system or activation of system built-in functionality (*e.g.*, RDP), which permits external persistent connections. After the establishment of a persistent connection, the attacker can start executing different actions while *moving laterally* in the environment. These actions leave system logs on end hosts, that we leverage in our host-based intrusion detection.

In addition to Command & Control, the kill-chain identifies LM as a crucial attack behavior. LM includes credential stealing and infiltrating other hosts controlled by attackers, to move laterally within the network and gain higher privileges to fulfill adversarial objectives. Fig. 1 illustrates an example of LM. In this figure, a host (*i.e.*, Host 1) that resides in an enterprise network is compromised by an attacker via social engineering, such as (spear) phishing [16]. Suppose there was a previous RDP connection from Host 1 to Host 2, and the credential used for accessing Host 2 is cached on Host 1. In this case, the attacker can perform credential stealing on Host 1 to gain access to another internal host (*i.e.*, LM to Host 2) that has physical access to the databases. Note that these databases are not directly connected to the Internet. The attacker can then make connection attempts to the internal databases using the stolen credentials of another internal host. Indeed, it is less likely that adversaries could launch a successful intrusion without LM, as crucial assets are typically not directly reachable from the outside of a network [20]. Thus, detecting APT using LM can also contribute to early attack detection [19,21]. In this paper, we focus on host-based RDP evidence for LM detection.

2.2. Related works

Host-based intrusion detection enables quick microscopic per-host analysis, and is well-suited for observable malware activities. It is typically accomplished by examining system traces, such as event logs and system calls. Existing works [12,22] show that host-based detection has a higher potential in comparison to its signature-based counterpart. However, as it requires extensive monitoring of system activities, it tends to consume host resources (*e.g.*, CPU cycles, memory, virtual machines). Consequently, this can negatively impact user experience on the host. Therefore, we use event logs collected by the *native* Windows event monitoring system, to minimize this logging overhead.

While Windows event logs can be used for detecting anomalous RDP sessions, they are also useful in detecting malicious tools executed on end hosts. Berlin et al. [23] implement a virus detection system that complements the host anti-virus software by applying ML techniques on Windows event logs. Therefore, similar techniques can be useful to detect the execution of malicious tools used during LM. However, it is a challenge to achieve low error rates in host-based intrusion detection [22]. Nevertheless, host-based analysis for LM detection is advantageous over the network-based alternative with respect to granularity and scalability [24].

Ussath et al. [16] analyze techniques and methods employed in 22 different APT campaigns, and help reveal their different relevant

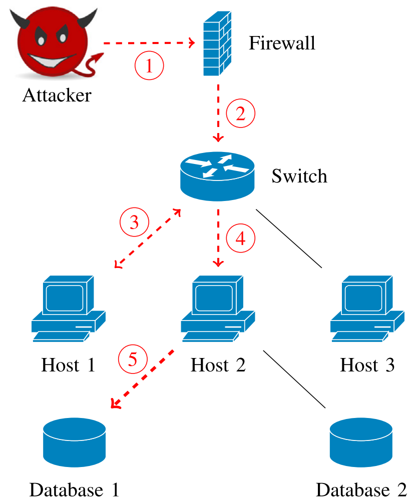

Fig. 1. An illustration of lateral movement.

characteristics. According to the authors, different tools and techniques are leveraged in different phases of APT attacks. Among these tools, RDP is one of the most popular technique for obtaining persistent access. Surprisingly, none of the surveyed APT campaigns use zero-day attacks during the LM phase. They also implement a user behavior simulation system [25] to generate user activity logs for Windows platforms. They leverage feed-forward and recurrent neural networks to identify malicious log events. However, their dataset, generated by simulation, is based on hypothetical assumptions. For example, ML features such as longitude and latitude of the user, are impractical for most real-world scenarios.

Kaiafas et al. [5] successfully employ an ensemble of classifiers for detecting malicious events in the LANL dataset [17]. The features used are extracted based on a constructed bipartite graph. However, the authors are oblivious to the biased nature of the LANL dataset. Based on our analysis (*cf.*, Section 3), all red team events in the dataset originate from *four* unique hosts. This implies that the ML classifiers will be biased to the source host feature (employed in [5]) in training and inference. We highlight this limitation in Section 5.

Siadati et al. [26] propose APT-Hunter that visualizes the logon connections between computers. By filtering out logon events specified by the security analysts, the unusual logon events can be further analyzed. However, such a system requires constant monitoring and filtering by the experts. The authors also implement a system [27] that extracts anomalous logon patterns. They propose a pattern mining algorithm that consists of two components, an exact matching classifier and a pattern matching classifier. While the exact matching classifier is prone to logon history poisoning, the pattern matching classifier complements it by matching a logon to all possible combination of attributes that describe it. A real dataset provided by a financial institution is employed for evaluation. However, due to the lack of malicious activities, they inject attack traces based on pen test campaigns. Their system yields 82% recall and 99.7% precision in detecting malicious logons. While the authors propose host-based detection that leverages pattern matching, the focus of our work is to harness ML techniques for intrusion detection.

Milajerdi et al. [28] develop a system, called Holmes, that leverages correlation between suspicious flows during an APT attack. It aims to map suspicious events found in the host logs to stages of an APT attack. To achieve this goal, Holmes first constructs a high-level scenario graph by mapping low level audit logs to behavioral patterns defined as Tactics, Techniques, and Procedures (TTPs). These TTPs are patterns from commonly used techniques in APT attacks. Then, it maps a set of TTPs to a particular stage in an APT attack. The proposed system is evaluated on a dataset generated from engagements of red teams and blue teams. This dataset contains 9 different APT scenarios and Holmes is able to achieve 100% recall and precision by selecting the optimal threshold for malicious scores. The main limitation of this work is the patterns used for generating TTPs, which require constant updates in order to detect new threats. Notably our system does not depend on any database to perform classification.

Lopez and Sartipi [29] propose different feature extraction techniques and provide a list of features that can be employed for detecting Information System misuse. The authors employ logistic regression on the LANL dataset. Their Receiver Operating Characteristic (ROC) produces a 82% area under the ROC curve, which outperforms random draw. Although, the authors propose features for detecting misuse, they do not evaluate the performance of various ML techniques on these features.

Creech et al. [22] design a host-based intrusion detection system that leverages system call patterns. They use a new type of neural network *i.e.*, extreme learning machine, with novel features derived from semantic analysis to achieve a high detection rate. They employ two datasets (*i.e.*, KDD98 and ADFA-LD) for evaluation with 100% detection rate and 0.6% false alarm rate. While their approach relies on the analysis of system calls, our work focuses on log analysis. In addition, their solution is customized for Linux-based systems, making it infeasible to directly leverage on the Windows platform due to the inherent differences in OS architectures [30].

Biggio et al. [14] analyze pattern recognition systems under adversarial settings. The authors point out that the existing pattern recognition systems are designed without taking security into consideration. Once the underlying assumption of data stationarity is broken, malicious attackers are capable of easily compromising the classifier. They review numerous existing works that leverage ML, and highlight vulnerability with examples and experiments. To cope with the security vulnerability in the clustering and classification systems, the authors propose both proactive and reactive defense approaches. The proactive approach can be categorized into security by design and security by obscurity, whereas the reactive approach focuses on learning from the past. This work helps us outline the different types of adversarial attacks that may compromise our proposed model.

Apruzzese et al. [31] study a network-based intrusion detection system [32] that uses ML techniques. The analyzed system uses network flow-based features with a random forest classifier for detecting botnet. The CTU-13 [33] dataset is used for evaluation. According to the experiments performed by the authors, botnet can easily evade the detection of such a classifier by slightly modifying its original commutation patterns (*e.g.*, flow duration, source bytes, destination bytes and total packets, *etc.*). The authors further demonstrate the effect of perturbing different combination of features. For instance, the detection rate of botnet drops from 99.85% to 19.22% after adding just 1 second to flow duration. While they analyze the weakness of a network-based intrusion detection system, we develop a similar benchmark approach to show that our model is robust against this type of adversarial attacks.

3. Dataset

The dataset plays a crucial role in the success of ML. However, Windows event log datasets that represent real user behavior are fairly limited. Most publicly available datasets, such as [33,34], facilitate network-based intrusion detection. In contrast, host event logs contain sensitive information limiting their distribution by organizations [25]. To overcome this limitation, researchers (*e.g.*, [25]) often simulate user and attacker behavior to generate synthetic datasets. However, datasets generated using this approach are purely based on hypothetical assumptions, and may not depict real-world user behavior. Therefore, to preserve the realism of user behavior, we leverage and combine two real datasets from LANL, namely *comprehensive* and *unified* datasets. In the combined dataset (*cf.*, Section 4), the Windows event IDs of interest to this work are 4624, 4625 and 4634, which pertain to RDP authentication. Table 1 provides a description of these events.

Table 1
Windows event ID Ref. [18].

Event ID	Description
4624	An account was successfully logged on
4625	An account failed to log on
4634	An account was logged off

3.1. Comprehensive events dataset

The comprehensive dataset [17] spans 58 days, and consist of activities generated from 12,425 users and 17,684 computers. The dataset is divided into five different logs, namely authentication, process, flow, DNS and red team logs. The red team log contains a subset of events from the authentication log, which are generated from red team activities (*e.g.*, compromise events). Hence, the red team log provides the ground truth for ML. In this paper, we leverage the authentication and red team logs for detecting malicious RDP sessions. However, based on the dataset description and our observations, there are limitations in the authentication log:

- The number of red team events is very small, accounting for less than 0.0001% of the total events and only appear in certain time intervals.
- There are no logoff events, making it impossible to deduce certain crucial features, such as the logon session duration.
- The timestamp is obfuscated in UNIX time epoch. As a result, it is difficult to categorize events into days, which could be a discriminating feature to identify abnormal usage.
- A large number of RDP logon events have the same source and destination host, which is beyond reason.

3.2. Unified events dataset

The unified dataset [18] is collected within LANL over a 90 day interval. Table 2 highlights a sample event from this dataset. Unlike the previous dataset, this dataset provides comprehensive and detailed Windows event logs, including the missing logoff events. Although the timestamps in this dataset are also obfuscated, events are already divided into days. However, the primary limitation of this dataset is the lack of red team activities, *i.e.*, this dataset only contains benign user activities. Furthermore, the source host is missing in some 4624 LogonType 10 events and all 4625 LogonType 10 events. The 4624 event records all successful logons and event 4625 records logon failures with reason, while type 10 in both events indicate that RDP is used for remote login. Both of these events are crucial for tracking (malicious) RDP sessions [35].

4. Methodology

4.1. Combining datasets

Both datasets have limitations according to their authors [36] and our observations. Hence we decided to inject red team events from the comprehensive dataset [17] into the unified dataset [18]. Since these two datasets were collected within the same organization, we do not lose the properties and patterns of attack events. However, these two datasets are obfuscated with different hash functions and cannot be simply merged. Also, recall that the red team events originate from only four unique hosts. Indeed we could have mapped these four source hosts into a larger group of hosts in our synthetic dataset to avoid any bias in the ML classifier. However, we did not choose this approach to preserve the authenticity of the attacks.

We ensure that: (i) the network topology and the communication patterns between benign hosts, along with the attack patterns are not modified, and (ii) the synthetic field in the dataset (*i.e.*, session duration), uses a statistical (*i.e.*, normal) distribution. Let R be the collection of red team logon events from the comprehensive dataset and B the collection of benign RDP logon events extracted from the unified dataset. For each event $e_i \in R$, we map the source host Src_i to a randomly selected unique source host Src_j from an event $e_j \in B$. We further map the user name and destination host tuple $\{Usr_i, Dst_i\}$ of e_i, to a randomly selected unique tuple $\{Usr_k, Dst_k\}$ from an event $e_k \in B$. After mapping, we insert, in chronological order, the modified red team events e_i' into the set B, labeled as *malicious*. There are no changes needed for timestamp, since the unified dataset already spans the red team events time interval (*i.e.*, the normal events span 90 days and red team events span first 30 days). The detailed injection algorithm is depicted in Algorithm 1.

We extract a total of 222,692 events with IDs 4624, 4625 and 4634, and authentication type 10. We discard all 4625 events and those 4624 events with missing source host. After cleaning the dataset of invalid data entries and extracting relevant features (*cf.*, Section 5), we end up with 56,837 events. The significant reduction in datapoints comes from combining logon events (ID 4624) with their corresponding logoff event (ID 4634) into an RDP session event with a well-defined session length. Benign logon events from the unified dataset with no corresponding logoff events are omitted as well.

Note that the injected red team authentication events only contain logon events (ID 4624) but no logoff events (ID 4634). Hence, this hampers the computation of malicious RDP session's duration. To this end, we generate a session duration for each red team event from a normal distribution $\mathcal{N}(\mu, \sigma^2)$, where μ and σ are the mean and standard deviation, respectively, computed from all benign RDP session's duration. Though a random distribution may be more reasonable, as attacks can last for any duration, we assume that attacks have similar behavior (session duration) to benign users. This assumption makes the classification problem more difficult, since the malicious data points are closer to benign data points in terms of this feature. It also makes the data more realistic, as some attackers may simulate the benign activities to avoid detection.

4.2. Feature engineering

We extract the following baseline features from the combined dataset derived in the previous subsection:

- *User (Usr)*: The user name used for RDP authentication.
- *Source (Src)*: The source host where the RDP authentication originated.
- *Destination (Dst)*: The destination host for the RDP authentication.
- *Session duration*: The duration of the RDP session in seconds.
- *User time difference*: For user Usr_i, the time difference of two sequential RDP authentication events e_j and e_k that contain user Usr_i.
- *Source time difference*: For source host Src_i, the time difference of two sequential RDP authentication events e_j and e_k that contain host Src_i.
- *Destination time difference*: For destination host Dst_i, the time difference of two sequential RDP authentication events e_j and e_k that contain host Dst_i.
- *Mean of session duration for user*: The average duration of all RDP sessions that contain user Usr_i.
- *Mean of session duration for source*: The average duration of all RDP sessions that contain source host Src_i.
- *Mean of session duration for destination*: The average duration of all RDP sessions that contain destination host Dst_i.
- *Weekday*: The weekday extracted from timestamp.
- *Seconds in a day*: The seconds elapsed within a day.

Not all the attributes from the original dataset (*cf.*, Table 2) are employed to extract the above features. Features such as event ID, process name, process ID, logon type description and domain name have identical values across all events. Therefore, we remove them from our feature list. The logon ID is used to compute session duration only.

Furthermore, we do not employ the timestamp as is, but instead we extract from the timestamp the weekday and the time (in seconds) in

Table 2
A sample event extracted from the unified dataset.

Field	Value	Description
UserName	User451666	User name used for authentication
EventID	4624	Microsoft defined Windows event ID
LogHost	Comp313779	The destination host that authentication targeted
LogonID	0 × 9c279eb	A semi-unique ID for current logon session
DomainName	Domain001	Domain name of user name
Source	Comp288750	The source host that authentication originated from
LogonType	RemoteInteractive	Description of logon type below
ProcessName	winlogon.exe	Process that processed the authentication event
Time	732	The obfuscated epoch time of the event in seconds
LogonType	10	Type of authentication event (*e.g.*, remote or local)
ProcessID	0xaa4	A semi-unique ID identifies process

Algorithm 1 Inject malicious RDP authentication events into benign events

Input: Benign RDP authentication events $Benign$, red team RDP events $Malicious$
Output: A synthetic dataset that combines benign and red team RDP events

```
/* Initialize some variables */
μ ← Benign.sessions.duration.mean()                ▷ Mean of session duration of Benign
σ^2 ← Benign.sessions.duration.variance()          ▷ Variance of session duration of Benign
RedHosts ← Malicious.sourceHosts                   ▷ Set of source hosts in Benign
BenignHosts ← Benign.sourceHosts                   ▷ Set of source hosts in Malicious
/* An authentication tuple is a combination of user name and destination host in an authentication event */
RedTuples ← Malicious.authTuples                   ▷ Set of authentication tuples in Benign
BenignTuples ← Benign.authTuples                   ▷ Set of authentication tuples in Malicious
/* Create a dictionary that maintains a one-to-one mapping from a original source host in Malicious to a newly selected host in Benign */
Source ← dict{}
for each host ∈ RedHosts do
    Source[host] ← BenignHosts.randomPop()
End
/* Create a dictionary that maintains a one-to-one mapping from a tuple (user name, destination host) in Malicious to a newly selected tuple in Benign */
AuthTuple ← dict{}
for each tuple ∈ RedTuples do
    AuthTuple[tuple] ← BenignTuples.randomPop()
End
/* Rewrite the fields in red team events and insert the modified event into Benign */
for each event ∈ Malicious do
    newSrc ← Source[event.SourceHost]
    newUser ← AuthTuple[event.AuthTuple].UserName
    newDst ← AuthTuple[event.AuthTuple].DestinationHost
    session ← GaussianRandom(μ, σ^2)
    modified ← newEvent(event.timeStamp, newSrc, newUser, newDst, session)
    Benign.append(modified)
End
return Benign
```

the day, which are more meaningful. Since timestamps are obfuscated, it is not straightforward to obtain weekday information directly by converting it from UNIX time epoch to date. Therefore, we leverage the count of RDP events per day to identify a pattern, as depicted in Fig. 2. That is, we identify the two consecutive days with least number of events in a 7 day interval as Saturday and Sunday.

4.3. ML techniques

4.3.1. Supervised learning algorithms

Based on previous studies [3,5,25,37], we select a variety of ML techniques that have proven effective in intrusion detection. We leverage Logistic Regression (LR), a classic regression model that is known to capture the relationship between variables. Similarly, we employ Gaussian-NB (GNB), a probabilistic classifier based on Bayes' theorem, without specifying any prior distribution. We also evaluate the DT classifier with a maximum depth of three and entropy criterion. The DT algorithm constructs a tree structure where each internal node splits data points based on pre-defined criterion. The DT used in our work is an optimized version of Classification and Regression Trees algorithm [38]. Furthermore, we evaluate Random Forest (RF) [39], LogitBoost (LB) [40] and LightGBM (LGBM) [41], which are ensemble methods built on top of DT. RF tends to solve the over-fitting problem in DT, whereas LB combines a set of weak learners to construct a strong learner. LGBM is similar to LB, and is a recent DT-based gradient boost algorithm. In comparison to other Gradient Boosting Decision Tree, the efficiency and scalability of LGBM is better by one order of magnitude [41]. We also evaluate Feed-forward Neural Network (FNN), a simple neural network without cycles between each layer.

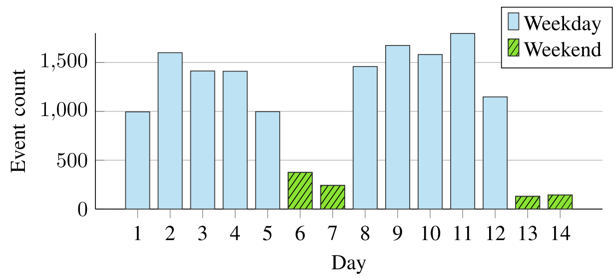

Fig. 2. RDP events per day.

4.3.2. Metrics

We define malicious RDP sessions as positive subjects and use the following performance metrics to evaluate the different ML techniques:

$$Accuracy = \frac{True\ Positive + True\ Negative}{Total\ number\ of\ subjects} \times 100\%$$

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

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

$$F_1\ score = 2 \times \frac{Recall \times Precision}{Recall + Precision}$$

$$AP\ score = \sum_n (Recall_n - Recall_{n-1}) \times Precision_n$$

The accuracy indicates the percentage of sessions that are correctly classified. Whereas, precision is the percentage of sessions that have been identified as malicious are indeed malicious. A higher precision implies a higher confidence in the true nature of the sessions flagged as malicious (*i.e.*, lower false positives). On the other hand, recall is the percentage of malicious sessions that have been correctly identified. A higher recall implies a higher confidence that malicious sessions are not missed (*i.e.*, lower false negatives). We also present the F_1 score, a harmonic mean of precision and recall. This metric provides the aggregate performance of a classifier. Though accuracy also depicts the overall performance, F_1 is more reliable when the dataset is imbalanced. In our case, the dataset used contains less than 3% of anomalous RDP sessions. Hence, a classifier could achieve superior accuracy (*e.g.*, more than 97% accuracy) by simply marking every RDP session as normal.

To illustrate the performance of the classifiers at different classification thresholds, we leverage the Precision–Recall (PR) curve. We also use the Average Precision (AP) score, which is the weighted average of precision at each decision threshold, and estimates the area under the PR curve. Although a ROC curve can also illustrate the aggregate performance of a classifier, it suffers with imbalanced datasets. Hence, we do not include it in our metrics.

5. Evaluation

5.1. Environment setup

5.1.1. Hardware

The data analysis, visualization and pre-processing are performed on a cluster of four nodes, each featuring an Intel(R) Xeon(R) E3-1230 v3 3.30 GHz CPU and 16 GB RAM. Nodes are interconnected with 10 Gbps Ethernet. Model training and validation are performed on an Amazon AWS EC2 t3.medium instance.

5.1.2. Software

A Logstash instance is deployed to ingest the dataset into an ElasticSearch [42] cluster, and Kibana is used for data visualization. For data pre-processing, a variety of Python packages, including Numpy [43], Scipy [44] and Pandas [45] are employed. The ML models are developed in Python with Scikit-learn [46] and Keras [47] libraries.

Table 3
RDP session detection with all baseline features.

Classifier	Accuracy	Precision	Recall	F_1
LR	98.50%	10.93%	1.74%	0.030
DT	99.90%	99.04%	93.58%	0.962
FNN	98.68%	0%	0%	0
GNB	99.60%	87.31%	82.11%	0.846
RF	99.95%	99.73%	96.13%	0.979
LB	99.99%	99.87%	99.73%	0.998
LGBM	99.99%	99.73%	99.33%	0.995

Table 4
Robustness of stand-alone RF in the face of unknown malicious *src* hosts.

Method	Accuracy	Precision	Recall	F_1
Cross-validation	99.50%	86.17%	74.10%	0.797
Robustness test	99.96%	0%	0%	0

5.2. Experiment

To validate our ML models, we first employ k-fold cross-validation ($k = 10$) with *all* baseline features, as depicted in Table 3. The FNN with three layers, 100, 50 and 1 neuron in each layer, respectively, and multiple activation functions (*i.e.*, sigmoid and ReLu), classifies all RDP sessions as benign. We tweaked the FNN by adjusting the number of layers, the number of neurons in each layer and the activation function, but to no avail. This can be attributed to the imbalanced nature of the dataset, as the malicious events only account for a small fraction of the total events (*cf.*, Section 3). Therefore, even though the FNN classifier has an outstanding accuracy of 98.68%, it results in zero precision and recall with all malicious RDP sessions misclassified as benign.

Although sampling techniques can be used to balance the dataset, they will cause other problems. In particular, the under-sampling algorithms are known to inherently lose critical information, while the over-sampling algorithms suffer from over-fitting [48]. Hence, we do not explore sampling techniques in this paper. Due to FNN's poor performance, we exclude it from the remaining evaluations. In contrast, the DT algorithms have both high precision and recall, with LB using DT regressor outperforming all other classifiers. This is primarily because LB classifiers are designed for boosting the performance of existing classifiers [40]. Another boosting algorithm, LGBM, achieves a slightly inferior performance than LB in precision and recall. Even though the probabilistic GNB classifier under performs the DT-based classifiers, it outperforms LR and FNN.

Recall that all the attacks in the employed dataset originate from four unique source hosts. Therefore, a classifier that uses the source host feature may tend to predict all events with these source hosts as malicious, leading to a bias in classification. To highlight this impact, we perform a robustness test with a RF classifier that leverages a subset of the original features *i.e.*, user name, source host, destination host, duration and timestamp. In this test, we demonstrate that even the simplest classifier with biased features can achieve excellent cross-validation results. However, such a classifier fails in detecting unknown attacks.

We split the dataset into training and testing sets, where the testing set contains malicious events that originate from source hosts that are not present in the training set. This allows us to run a *robustness* test. As shown in Table 4, this results in over-fitting, with RF unable to correctly classify any malicious RDP session from unknown source hosts.

Therefore, we remove features from our feature set that cause such a bias, namely username, source host and destination host. Table 5 depicts the result after the removal of these features. In comparison to Table 3, no significant difference is evident in terms of precision, recall or F_1 score in classifying RDP sessions. The LGBM classifier achieves perfect precision in this test case even though its overall performance decreases slightly. Although these results are promising, and most malicious RDP sessions are detected with low false positives, we attempt to further improve the performance of our ML models.

Table 5
RDP session classification (*user*, *src* and *dst* features removed).

Classifier	Accuracy	Precision	Recall	F_1
LR	98.50%	11.34%	1.87%	0.321
DT	99.90%	99.04%	93.58%	0.962
FNN	98.68%	0%	0%	0
GNB	99.60%	87.31%	82.11%	0.846
RF	99.94%	99.59%	96.13%	0.978
LB	99.99%	99.87%	99.47%	0.997
LGBM	99.99%	100%	98.8%	0.994

Table 6
Majority voting for RDP session detection using naïve approach *i.e.*, all five classifiers and baseline features.

Classifier	Accuracy	Precision	Recall	F_1
GNB, RF, LB, LR, DT, LGB	99.91%	99.86%	93.19%	0.964

Table 7
Majority voting for RDP session classification using selective classifiers (*user*, *src*, and *dst* features removed).

Classifier	Accuracy	Precision	Recall	F_1
GNB, RF, LB	99.95%	99.87%	96.26%	0.980
GNB, RF, DT	99.91%	99.58%	93.32%	0.964
GNB, LB, DT	99.91%	99.73%	93.32%	0.964
RF, LB, DT	99.95%	99.73%	96.13%	0.979

5.2.1. Voting

The authors in [5] improve the performance of their stand-alone classifiers by consolidating them using ensemble ML. We employ a similar approach with Majority Voting (MV) algorithm, starting with a naïve attempt that leverages all ML models in the ensemble. This results in a lower precision, recall and F_1 score, as shown in Table 6. Since weak classifiers can influence the voting process, this suggests a careful selection of the classifiers to include in the ensemble prior to applying MV. Therefore, due to the lackluster performance of LR and FNN (*cf.*, Table 5), we remove them from the ensemble. In addition, we also eliminate classifiers from the same category with relatively poorer performance (*i.e.*, DT is removed since RF has better performance, and LGBM is removed because of LogitBoost). The performance of the combined classifiers is shown in Table 7. In comparison to the previous ensemble depicted in Table 6, the classification of RDP sessions improve, but still under performs stand-alone LB in the best case. The best performing ensemble has minor improvements with respect to precision, but results in a much lower recall than the stand-alone LB classifier.

Evidently, MV is unable to boost the performance of the stand-alone classifiers. Therefore, we explore other ensemble approaches, namely Weighted Voting (WV) and its special-case Conservative Approach (CA). We assign weights based on intuition. A higher weight for the best classifier may reduce both false positives and false negatives. Whereas, a low threshold with equal weight could improve the true positives. We select the best performing ensemble from Table 7. The first three columns in Table 8 are the weights assigned to each classifier, namely LB, RF and GNB. The threshold is the ratio of votes required for a RDP session to be classified as malicious. For example, the second row in the table assigns LB a weight that is equal to the sum of weights for the remaining two classifiers. Intuitively, this has the potential to identify more true positives (*i.e.*, malicious RDP sessions) that are missed by LB. However, this combination is unable to spot any extra malicious sessions, as shown in Table 8. In this case, the malicious sessions identified by RF and GNB have already been recognized by LB.

In the last row of the table, a RDP session is classified as malicious if any classifier in the ensemble tags it as malicious, which corresponds to CA. Though this results in a slight increase in recall, it comes at the cost of a large drop of precision in classifying RDP sessions and yields a lower F_1 score. Therefore, we choose the stand-alone LB classifier

Table 8
Weighted voting for RDP session classification using LB, RF and GNB classifiers (*user*, *src*, and *dst* features removed).

LB	RF	GNB	Thr.	Accuracy	Precision	Recall	F_1
0.25	0.25	0.25	0.5	99.95%	99.87%	96.26%	0.980
0.5	0.25	0.25	0.5	99.99%	99.87%	99.47%	0.997
0.25	0.25	0.25	0.25	99.83%	89.03%	99.60%	0.940

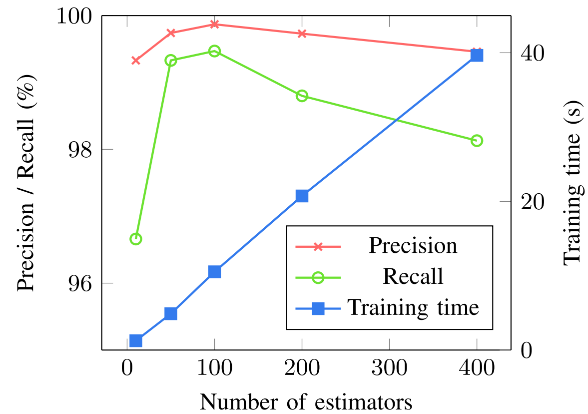

Fig. 3. Recall, precision and training time *vs.* # of estimators for stand-alone LB (our model).

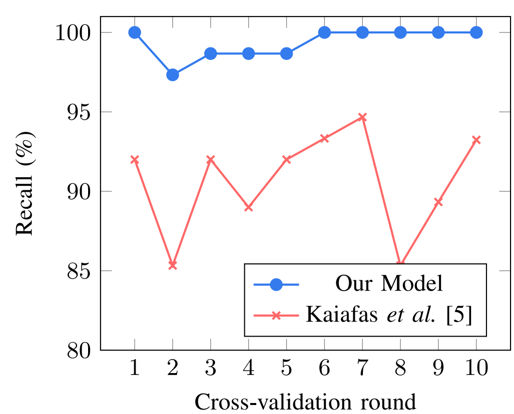

Fig. 4. Recall during each cross-validation round.

as *Our Model* for comparison to the state-of-the-art. This LB classifier uses DT as the base estimator, where the number of estimators can significantly impact performance. The training time increases linearly with the number of estimators, as shown in Fig. 3, while precision and recall are the highest with around 100 estimators. In the remaining experiments, we use stand-alone LB with 100 estimators.

5.2.2. Comparative analysis

We compare our stand-alone LB classifier with Kaiafas et al. [5]. The LB classifier is preferred over the LGBM because we do not want to miss any attack. Although LGBM achieves perfect precision in our previous experiment (*cf.*, Table 5), its recall is lower than LB. As mentioned in Section 1, our goal in this paper is to optimize the recall. In other words, we tolerate a higher number of false positives in exchange for a lower number of false negatives.

In order to compare, we implement Kaiafas' [5] approach and evaluate the corresponding model on our dataset. During feature extraction, we omit their geometric distribution feature, since all the failure events are filtered out due to missing source host in the dataset. As shown in Table 9, with all available features, the recall of Kaiafas' model is slightly lower than our model (first two rows without a *). Though

Table 9
RDP session classification using stand-alone LB vs. [5].

Classifier	Accuracy	Precision	Recall	F_1	TT (s)
Our Model	99.99%	99.87%	99.73%	0.998	11.28
Kaiafas et al.	99.98%	100.00%	98.67%	0.993	20.48
[a]Our Model	99.98%	99.87%	99.47%	0.992	10.53
[a]Kaiafas et al.	99.88%	100.00%	90.66%	0.951	18.19

[a]= Model validation without *user*, *src* and *dst* features.

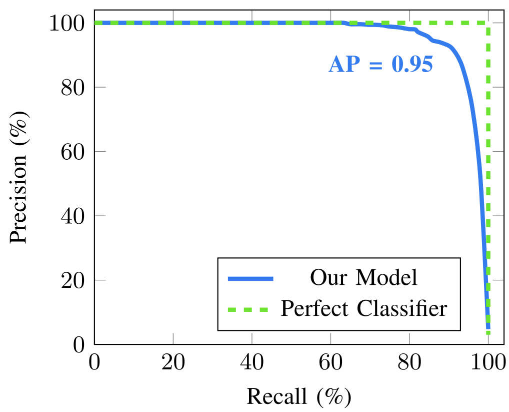

Fig. 5. Precision–recall curve of our model.

their precision is better, the F_1 score indicates an overall performance drop in comparison to our model. After the removal of user name, source host and destination host features from both models, Kaiafas' model has a significant drop in recall from 98.67% to 90.66% (last two rows with a *). In addition, the standard deviation of recall is high for [5]. For some rounds of cross-validation, it can only achieve 85% recall, as shown in Fig. 4. On the other hand, our model illustrates stability in recall over multiple cross-validation rounds. Furthermore, the training time (TT) of Kaiafas' model is about 80% higher than our model. This can be primarily attributed to the larger number of features and construction of extra classifiers. Therefore, our model outperforms a state-of-the-art in RDP session classification in terms of both performance and training time.

Finally, to further evaluate the models against zero-day threats, we perform a robustness test. We split the dataset into training (75%) and testing (25%). While the training set contains attacks originating from three different sources, the testing set contains an additional attacking source that does not appear in the training set. In Fig. 5, we present the PR curve, which illustrates the trade-off between precision and recall at different thresholds. As evident, our model's PR curve is very close to a perfect classifier and yields an AP score of 0.95. This asserts the robustness of our model to detect threats from new (unseen) attack sources. However, it is unfeasible to plot a PR curve for a MV classifier, such as Kaiafas' model. A MV classifier depends on decisions made by several classifiers and a single chosen threshold across classifiers in not appropriate. Therefore, we compare the overall performance of the two classifiers using the F_1 score. While our model achieves the highest F_1 score of 0.914, Kaiafas' model scores a low 0.675.

5.2.3. Robustness to adversarial attempts

ML algorithms were originally designed without considering adversaries that may intentionally fabricate the input data to manipulate the outcome of a classifier [49,50]. Typically, they assume a benign environment, where both training and testing datasets are stationary, and follow the same statistical distribution. According to [15], adversarial attacks can be categorized along three axes (*i.e.*, attack influence, attack specificity and security violation). A brief description of these axes is presented in Table 10. In addition, we list examples of potential adversarial attacks against our classification model in Table 11, which are inspired from Table 10. We focus on exploratory attacks, with the assumption that attackers do not have access to the classifier and training dataset. Indeed, should the adversaries obtain access to the training dataset (or the classifier itself), they can easily mimic a benign user's logon pattern by learning their authentication and communication patterns. Since our model heavily depends on the benign user behavior, it will perform poorly under such circumstances. In addition, this can potentially undermine the integrity of any intrusion detection approach.

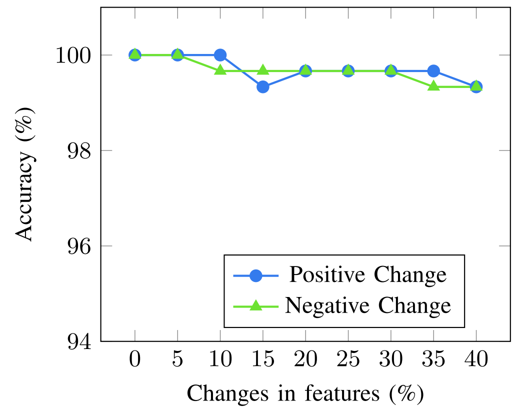

Fig. 6. Detection accuracy for polymorphic forms of *known* attack.

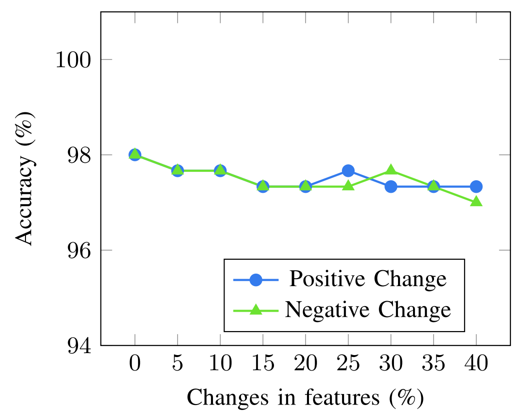

Fig. 7. Detection accuracy for polymorphic forms of *unknown* attack.

In order to study the impact of adversarial attacks against our model, we conduct a series of experiments. Under previous assumptions, we create new RDP sessions with polymorphic form of attacks by manipulating the features of a malicious RDP session. Potentially, this will impact the accuracy of our model, since it effectively alters the distribution of malicious data points. Not all of the features are modified in this process. In general, we do not perturb statistical features (*i.e.*, mean of session duration for user, mean of session duration for source and mean of session duration for destination), since they represent the existing user behavior pattern. We replace the original timestamp from selected malicious RDP sessions with a randomly generated timestamp. For the remaining features, we perturb them in percentages. An example of an adversarial sample is shown in Table 12.

The testing dataset in subsequent experiments only contains malicious RDP sessions. Thus, the precision is always 100%, while the accuracy is always equal to recall. Therefore, we use accuracy as the evaluation metric in these experiments. In the first experiment, we train our model on the entire dataset from Section 4 and create 300 new malicious RDP sessions for testing, by employing the aforementioned perturbations. Specifically, we randomly select 300 malicious

Table 10
Taxonomy of attacks against ML systems [49,50].

Attack influence	Exploratory	Attacker can only manipulate the testing data
	Causative	Attacker can manipulate both training and testing data
Attack specificity	Targeted	Attacker focuses on a subset of samples
	Indiscriminate	Attacker focuses on any sample
Security violation	Confidentiality	Attacker obtains confidential information
	Integrity	Attacker gains access to a restricted service or resource
	Availability	System denies legitimate access for benign users

Table 11
Examples of potential attacks against our ML model.

Exploratory	Targeted	Integrity	Attacker obfuscates RDP access pattern to simulate a particular benign user
	Indiscriminate	Integrity	Attacker obfuscates RDP access pattern to simulate any arbitrary benign user
Causative	Targeted	Integrity	Mis-train classifier to grant attacker RDP access to a single protected host
		Availability	Mis-train classifier to block benign RDP access of a particular benign user
	Indiscriminate	Integrity	Mis-train classifier to grant attacker RDP access to any protected host
		Availability	Mis-train classifier to block benign RDP access of all benign users

Table 12
Example of a polymorphic attack.

Data	Weekday	Seconds in a day	Session duration	User time diff	Source time diff	Destination time diff
Original	2	12 340	100	1000	1000	1000
Mutation	6 (random)	45 670 (random)	125 (+25%)	1250 (+25%)	1250 (+25%)	1250 (+25%)

data points from the training set and mutate them. The positive or negative change implies that the original feature values are increased or decreased by certain percentages, respectively.

The classification result of newly crafted RDP sessions is illustrated in Fig. 6. The steady lines indicate that our model is robust to polymorphic forms of *known* attacks. To further investigate the robustness of our model, the second experiment excludes randomly selected 300 malicious RDP sessions from the original training dataset. We apply the same perturbation approach on these malicious RDP sessions and classify them. The results of the second experiment is presented in Fig. 7. The overall performance drops by 2% in comparison to the first experiment. However, this is expected, since the classifier is handling polymorphic forms of *unknown* attacks. The plots from Fig. 7 have a similar trend to the plots in the previous figure. Both experiments indicate that our model is robust enough to exploratory types of adversarial attacks. This can be attributed to the success of capturing user's behavior within the features and the choice of ML classifier.

6. Conclusion and future work

RDP is one of the major tools employed during the lateral movement stage of an APT attack. Therefore, we leverage Windows event logs for detection of malicious RDP sessions. With the identified shortcomings of two public datasets, we synthesize a combined dataset that remains faithful to the attack models. Using the combined dataset, we extracted relevant features, and explore supervised learning algorithms to detect anomalous RDP sessions. After evaluating various classification algorithms, we chose LB as the best model with respect to accuracy, recall and precision in Windows RDP session classification. LB shows promising results and outperforms a state-of-the-art model [5] in recall and training time. In addition, we demonstrate that our approach is robust to adversarial attacks. In the future, we will evaluate our approach on other session-based protocols, such as Secure Shell. In addition, the Windows event logs contain a variety of event types, which can be leveraged to identify different stages of an APT attack. Therefore, we will also explore system events other than authentication to classify between benign and unauthorized use of system administration tools.

CRediT authorship contribution statement

Tim Bai: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing - original draft, Writing - review & editing, Visualization. **Haibo Bian:** Conceptualization, Methodology. **Mohammad A. Salahuddin:** Conceptualization, Methodology, Writing - review & editing, Visualization, Project administration. **Abbas Abou Daya:** Conceptualization, Methodology. **Noura Limam:** Conceptualization, Methodology, Writing - review & editing, Visualization, Project administration. **Raouf Boutaba:** Resources, Supervision, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported in part by the Royal Bank of Canada and in part by the NSERC CRD, Canada Grant No. 530335.

References

[1] Kaspersky Lab, Carbanak APT: The great bank robbery, 2015, accessed: 2020-11-04. [Online]. Available: https://securelist.com/the-great-bank-robbery-the-carbanak-apt/68732/.

[2] B. Krebs, Anthem breach may have started in april 2014, 2015, accessed: 2020-11-04. [Online]. Available: https://krebsonsecurity.com/2015/02/anthem-breach-may-have-started-in-april-2014/.

[3] R. Boutaba, M.A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano, O.M. Caicedo, A comprehensive survey on machine learning for networking: evolution, applications and research opportunities, J. Internet Serv. Appl. 9 (1) (2018).

[4] S. Ayoubi, N. Limam, M.A. Salahuddin, N. Shahriar, R. Boutaba, F. Estrada-Solano, O.M. Caicedo, Machine learning for cognitive network management, IEEE Commun. Mag. 56 (1) (2018) 158–165.

[5] G. Kaiafas, G. Varisteas, S. Lagraa, R. State, C.D. Nguyen, T. Ries, M. Ourdane, Detecting malicious authentication events trustfully, in: Proceedings of NOMS, 2018.

[6] G. Kim, S. Lee, S. Kim, A novel hybrid intrusion detection method integrating anomaly detection with misuse detection, Expert Syst. Appl. 41 (4) (2014) 1690–1700.

[7] A. Vance, Flow based analysis of Advanced Persistent Threats detecting targeted attacks in cloud computing, in: Proceedings of Intl. Scientific-Practical Conf. Problems of Infocommunications Sc. and Tech., 2014.

[8] M. Marchetti, F. Pierazzi, M. Colajanni, A. Guido, Analysis of high volumes of network traffic for advanced persistent threat detection, Comput. Netw. 109 (2016) 127–141, Traffic and Performance in the Big Data Era. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128616301633.
[9] S. Siddiqui, M.S. Khan, K. Ferens, W. Kinsner, Detecting advanced persistent threats using fractal dimension based machine learning classification, in: Proceedings of the ACM International Workshop on Security and Privacy Analytics, 2016.
[10] A.A. Daya, M.A. Salahuddin, N. Limam, R. Boutaba, A graph-based machine learning approach for bot detection, 2019.
[11] Y. Tsuda, J. Nakazato, Y. Takagi, D. Inoue, K. Nakao, K. Terada, A lightweight host-based intrusion detection based on process generation patterns, in: Proceedings of Asia Joint Conf. on Info. Security, 2018.
[12] D. Moon, S.B. Pan, I. Kim, Host-based intrusion detection system for secure human-centric computing, J. Supercomput. 72 (7) (2016).
[13] F.S. Team, As the Holiday Season Draws Near, Mobile Malware Attacks Are Prevalent, Fortinet, 2018.
[14] B. Biggio, G. Fumera, F. Roli, Pattern recognition systems under attack: Design issues and research challenges, Int. J. Pattern Recognit. Artif. Intell. 28 (07) (2014) 1460002.
[15] M. Barreno, B. Nelson, A.D. Joseph, J.D. Tygar, The security of machine learning, Mach. Learn. 81 (2) (2010) 121–148.
[16] M. Ussath, D. Jaeger, F. Cheng, C. Meinel, Advanced persistent threats: Behind the scenes, in: Proceedings of Annual Conference on Information Science and Systems (CISS), 2016.
[17] A.D. Kent, Comprehensive, Multi-Source Cyber-Security Events, Los Alamos National Laboratory, 2015.
[18] M.J.M. Turcotte, A.D. Kent, C. Hash, Unified host and network data set, in: Data Science for Cyber-Security, World Scientific, 2018, ch. Chapter 1.
[19] E.M. Hutchins, M.J. Cloppert, R.M. Amin, Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains, in: Leading Issues in Information Warfare & Security Research, Vol. 1, 2011.
[20] J.R. Vacca, Network and System Security, Elsevier, 2013.
[21] S. Khattak, N.R. Ramay, K.R. Khan, A.A. Syed, S.A. Khayam, A taxonomy of botnet behavior, detection, and defense, IEEE Commun. Surv. Tutor. 16 (2) (2014).
[22] G. Creech, J. Hu, A semantic approach to host-based intrusion detection systems using contiguousand discontiguous system call patterns, IEEE Trans. Comput. 63 (4) (2014).
[23] K. Berlin, D. Slater, J. Saxe, Malicious behavior detection using windows audit logs, in: Proceedings of the ACM Workshop on Artificial Intelligence and Security (AISec), 2015.
[24] H.-J. Liao, C.-H.R. Lin, Y.-C. Lin, K.-Y. Tung, Intrusion detection system: A comprehensive review, J. Netw. Comput. Appl. 36 (1) (2013) 16–24.
[25] M. Ussath, D. Jaeger, F. Cheng, C. Meinel, Identifying suspicious user behavior with neural networks, in: Proceeding of IEEE Intl. Conf. on Cyber Security and Cloud Computing, 2017.
[26] H. Siadati, B. Saket, N. Memon, Detecting malicious logins in enterprise networks using visualization, in: 2016 IEEE Symposium on Visualization for Cyber Security (VizSec), 2016, pp. 1–8.
[27] H. Siadati, N. Memon, Detecting structurally anomalous logins within enterprise networks, in: Proceedings of the ACM Conference on Computer and Communications Security, 2017, pp. 1273–1284.
[28] S.M. Milajerdi, R. Gjomemo, B. Eshete, R. Sekar, V. Venkatakrishnan, HOLMES: Real-time APT detection through correlation of suspicious information flows, in: 2019 2019 IEEE Symposium on Security and Privacy (SP), IEEE Computer Society, Los Alamitos, CA, USA, 2019, pp. 447–462.
[29] E. Lopez, K. Sartipi, Feature engineering in big data for detection of information systems misuse, in: Proceedings of the Annual Intl. Conf. on Computer Science and Software Engineering, 2018.
[30] G. Creech, Developing a High-Accuracy Cross Platform Host-Based Intrusion Detection System Capable of Reliably Detecting Zero-Day Attacks (Ph.D. dissertation), University of New South Wales, Canberra, Australia, 2014.
[31] G. Apruzzese, M. Colajanni, Evading botnet detectors based on flows and random forest with adversarial samples, in: 2018 IEEE 17th International Symposium on Network Computing and Applications (NCA), IEEE, 2018, pp. 1–8.
[32] M. Stevanovic, J.M. Pedersen, An analysis of network traffic classification for botnet detection, in: 2015 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA), IEEE, 2015, pp. 1–8.
[33] S. García, M. Grill, J. Stiborek, A. Zunino, An empirical comparison of botnet detection methods, Comput. Secur. 45 (2014).
[34] E.B.B. Samani, H.H. Jazi, N. Stakhanova, A.A. Ghorbani, Towards effective feature selection in machine learning-based botnet detection approaches, in: IEEE Conference on Communications and Network Security, 2014, pp. 247–255.
[35] JPCERT Coordination Center, Detecting lateral movement through tracking event logs, 2017, accessed: 2020-11-04. [Online]. Available: https://www.jpcert.or.jp/english/pub/sr/ir_research.html.
[36] A.D. Kent, Proceedings of cybersecurity data sources for dynamic network research, in: Dynamic Networks in Cybersecurity, 2015.
[37] C.-F. Tsai, Y.-F. Hsu, C.-Y. Lin, W.-Y. Lin, Intrusion detection by machine learning: A review, Expert Syst. Appl. 36 (10) (2009).
[38] L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone, Classification and Regression Trees, in: The Wadsworth Statistics/Probability Series, Wadsworth & Brooks/Cole Advanced Books & Software, Monterey, CA, 1984.
[39] T.K. Ho, Random decision forests, in: Proceedings of 3rd International Conference on Document Analysis and Recognition, Vol. 1, IEEE, 1995, pp. 278–282.
[40] J. Friedman, T. Hastie, R. Tibshirani, Additive logistic regression: a statistical view of boosting, Ann. Statist. 28 (1998) 2000.
[41] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, T.-Y. Liu, Lightgbm: A highly efficient gradient boosting decision tree, in: Advances in Neural Information Processing Systems, 2017, pp. 3146–3154.
[42] C. Gormley, Z. Tong, Elasticsearch: The Definitive Guide, first ed., O'Reilly Media, Inc., 2015.
[43] T.E. Oliphant, Guide to NumPy, second ed., CreateSpace Independent Publishing Platform, USA, 2015.
[44] E. Jones, T. Oliphant, P. Peterson, et al., SciPy: Open source scientific tools for Python, 2001–, accessed: 2020-11-04. [Online]. Available: http://www.scipy.org/.
[45] W. McKinney, Data structures for statistical computing in python, in: Proceedings of the Python in Science Conference, 2010.
[46] Pedregosa, et al., Scikit-learn: Machine learning in python, J. Mach. Learn. Res. 12 (2011).
[47] F. Chollet, et al., Keras, 2015, https://keras.io.
[48] P. Baldi, Autoencoders, unsupervised learning and deep architectures, in: Proceedings of the International Conference on Unsupervised and Transfer Learning Workshop, 2011, pp. 37–50.
[49] M. Barreno, B. Nelson, R. Sears, A.D. Joseph, J.D. Tygar, Can machine learning be secure? in: Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security, ACM, 2006, pp. 16–25.
[50] L. Huang, A.D. Joseph, B. Nelson, B.I. Rubinstein, J. Tygar, Adversarial machine learning, in: Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence, ACM, 2011, pp. 43–58.

Tim Bai is a candidate of MMath at the David R. Cheriton School of Computer Science, University of Waterloo. He received his BCS from University of Waterloo in 2017. His current research interests include the network softwarization, cybersecurity, and machine learning.

Haibo Bian is a candidate of MMath at the David R. Cheriton School of Computer Science, University of Waterloo. He received his BSE from Zhejiang University in 2016. His current research interests include the network function virtualization, cybersecurity, and machine learning.

Mohammad A. Salahuddin is a research assistant professor of computer science at the University of Waterloo. He received his Ph.D. in computer science from Western Michigan University in 2014. 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 is a reviewer for various journals and magazines.

Abbas Abou Daya received his MMath in Computer Science from the University of Waterloo in 2019. He has graduated from the American University of Beirut with a B.Eng. in Electrical and Computer Engineering. He is a senior software engineer at Arista Networks. His current research interests involve machine learning, cybersecurity, networks and systems.

Noura Limam received her M.Sc. and Ph.D. degrees in computer science from the University Pierre & Marie Curie, Paris VI, in 2002 and 2007, respectively. She is currently a research assistant professor of computer science at 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.

Raouf Boutaba received his M.Sc. and Ph.D. degrees in computer science from the University Pierre & Marie Curie, Paris, in 1990 and 1994, respectively. He is a professor in the Cheriton School of Computer Science and Associate Dean, Research of the Faculty of Mathematics at the University of Waterloo, and holds an INRIA International Chair at INRIA Nancy. His research interests include network and service management, cloud computing, network virtualization, and network softwarization.