A Machine Learning Approach for RDP-based Lateral Movement Detection

by

Zhenyu Bai

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2019

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

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 cyberattack 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 thesis, we propose to detect evidence of LM using an anomaly-based approach that leverages Windows RDP event logs. We explore different feature sets extracted from these logs and evaluate various supervised and unsupervised machine learning (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 demonstrate that our model is robust against certain types of adversarial attacks.

Acknowledgements

First of all, I would like to thank my supervisor Prof. Raouf Boutaba. He guided me throughout the entire graduate study. I am highly grateful to him for giving me the opportunity to pursue research and work on a variety of topics. It was an honor for me to be his student.

Secondly, I would like to express my sincere gratitude to Prof. Noura Liman, Dr. Mohammad Ali Salahuddin and Haibo Bian for their continuous support in the development of this project. I would also thank Abbas Abou Daya for his contribution in the early stage of this project.

Last but not the least, many special thanks to Shihabur Rahman Chowdhury and Anthony for working with me during my early graduate study.

Dedication

This is dedicated to my parents for their unconditional love and support. I also want to thank all who helped and supported me during my graduate study.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

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. Stuxnet [33], an infamous attack on critical infrastructure, devastated Iran's nuclear program. According to Kaspersky Lab [36], 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 [44]. In this attack, Mivast malware [19] masqueraded as a VPN software, serving as a backdoor for the attacker.

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 protected 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).

There exists numerous works [23,32,48,50] that propose systems and models for detecting different stages of an APT. However, only a few of them specifically focus on attacks that

leverage the Remote Desktop Protocol (RDP) [1] during the LM phase. In addition, these works are usually evaluated on generated datasets with hypothetical assumptions.

1.2 Challenges and Research Opportunities

During the LM phase, attackers tend to use legitimate system tools, which make the detection of APT a challenging endeavor. APT detection has primarily been achieved via two approaches, signature-based [26], [68] and anomaly-based [14, 57] methods. Signature-based methods rely on known databases of attack signatures. They are vulnerable to zero-day attacks and are susceptible to lag in database updates. On the other hand, anomaly-based approaches establish a baseline of the normal behavior for the system, and flag any divergence or statistical deviations from the norm. Thus, anomaly-based methods are more robust to polymorphic form of attacks.

Machine Learning (ML) techniques have been widely used for anomaly-based APT detection [9], since ML is an ideal tool to automatically establish the normal behavior of a system [3]. There are some studies that utilize a single ML model, while others combine different learning techniques to form an ensemble or a hybrid model for anomaly detection. For instance, Kaiafas *et al.* [32] build an ensemble classifier that leverages voting mechanism, whereas Kim *et al.* [43] employ both SVM and DT to build a two-stage classification model. All of these techniques demonstrate great advances in anomaly detection system.

Anomaly-based APT detection methods generally rely either on network flow data [68], [18,48,58], or host system logs [63], [51] to uncover evidence of APT. Network-based anomaly 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 [9], 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 TLS [61]. 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 behaviour within network data.

On the other hand, host-based anomaly 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.*, Chapter 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 [8], both network-based and host-based anomaly detection systems suffer from attacks of adversaries. Therefore, any ML-based system must be designed with defense strategies against adversarial attacks. There are numerous works that attempt to tackle the aforementioned issues (*cf.*, Chapter 2). For example, Marco *et al.* [5] 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.* [8] develop systematic approaches to defend against these 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 [66], since discriminating between legitimate and malicious use of this tool is challenging. We surveyed nine distinct APT incidents [16, 17, 25, 34–38, 54] and five of them (over 50%) used RDP during the attack. Therefore, in this thesis, we detect anomalous RDP sessions based on evidence from host logs with a focus on optimizing recall.

1.3 Contribution

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) [40, 64]. To overcome their limitations, we combine these two datasets while preserving their realistic properties.
- We propose an anomaly-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. In addition, we evaluate a model that combines unsupervised and supervised ML techniques to improve the performance of RDP session classification.

- We compare the performance of our proposed approach to a state-of-the-art method [32], 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.

1.4 Thesis Organization

The thesis is organized as follows:

- Chapter 2 provides background knowledge that is necessary to understand this thesis. In addition, it presents the current state of existing anomaly-based detection approaches.
- Chapter 3 describes the characteristics and properties of the two datasets we employed in this thesis.
- Chapter 4 presents the approach of crafting our own synthetic dataset based on existing dataset. Furthermore, the features extracted from this dataset are elaborated. In addition, ML techniques and their performance evaluation metrics are discussed in this chapter.
- Chapter 5 delineates the evaluation results of different approaches in detecting anomalous RDP sessions and benchmarks the robustness of the proposed model.
- Chapter 6 reviews our main contributions and provides a brief summary of this thesis. In addition, this chapter instigates future research directions.

Chapter 2

Background

2.1 Intrusion kill-chain and Lateral Movement

Conventional APT detection approaches assumes 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 [29], depicted in Fig. 2.1. 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 APT 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,

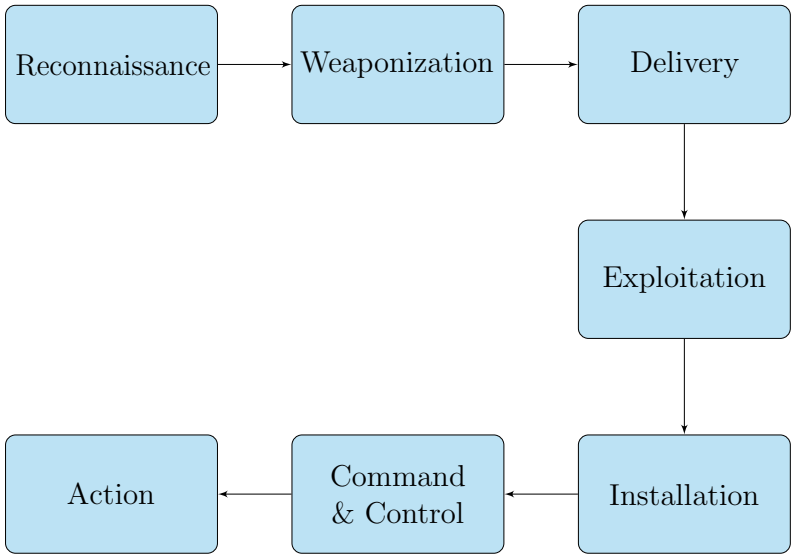

Figure 2.1: Intrusion kill-chain

to move laterally within the network and gain higher privileges to fulfill adversarial objectives. Fig. 2.2 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 [66]. 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. 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 [67]. Thus, detecting APT using LM can also contribute to early attack detection [29, 42]. In this thesis, we focus on host-based RDP evidence for LM detection.

2.2 Host-based Anomaly Detection

Host-based anomaly detection enables quick microscopic per-host analysis, and is well-suited for known and observable malware activities. It is typically accomplished by examining system traces, such as event logs and system calls. Existing works [14, 51] show that

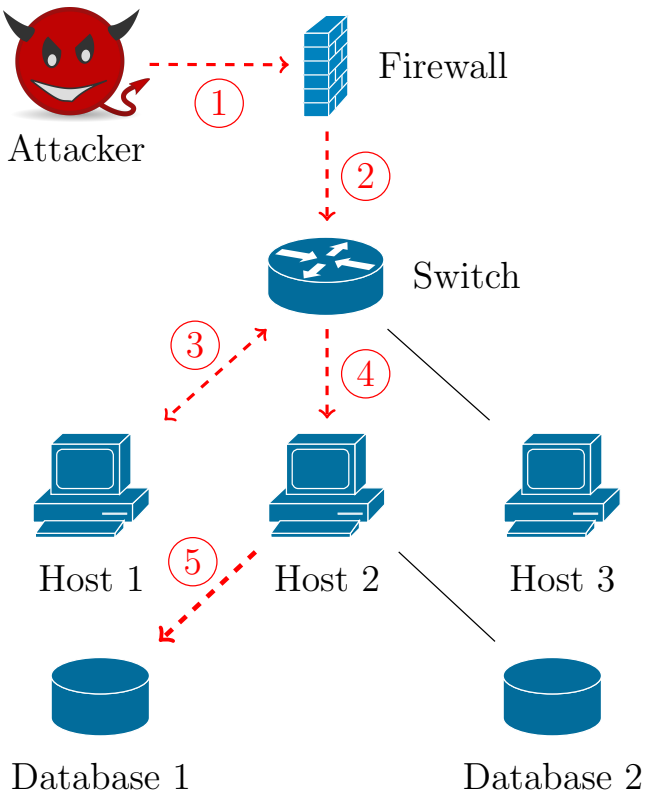

Figure 2.2: An illustration of LM

host-based anomaly 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. We use event logs collected by the *native* Windows event monitoring system, to minimize this overhead incurred during logging.

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.* [7] 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 anomaly detection [14]. Nevertheless, host-based analysis for LM detection is advantageous over the network-based alternative with respect to granularity and scalability [45].

2.3 Related Works

While this thesis focuses on a host-based approach for LM detection, we discuss comprehensive related works that help develop our approach. This section starts with the analysis of APT campaigns, which is followed by a discussion on related works by categories, namely host-based, network-based and hybrid approaches. We conclude with a few related works on adversarial machine learning.

2.3.1 Advanced Persistent Threats

Tankard [60] discusses the characteristics of APT attacks and identifies the fundamental difference between APT and conventional cyber threats. The author also analyzes a few APT attack cases with a focus on the various techniques employed during these attacks. There are also a few strategies proposed for helping organizations to defend against these APTs. For a host-based approach, the author suggests the adoption of Security Incident and Event Management (SIEM) to analyze host logs and identify anomalies. Whereas, network intrusion and detection systems are recommended for real time traffic monitoring in the network-based approach.

Ussath *et al.* [66] analyze techniques and methods employed in 22 different APT campigns, and help reveal different relevant characteristics of these campaigns. According to the authors, different tools and techniques are leveraged in different phases of APT attacks. During the LM stage, the majority of the campaigns use standard operating system tools to collect information and gain remote access. 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.

2.3.2 Network-Based APT Detection

Marchetti *et al.* [48] propose a framework that detects hosts that are involved in APT activities, specifically in data exfiltrations. The proposed framework starts with collecting network flows followed by feature extraction and normalization. Eventually, each internal host is assigned a suspicious score that is computed from extracted features. The prototype of this framework is deployed and evaluated in a real network that consists of around 10,000 hosts. The proposed system is scalable as it can process 140 million flows within 2 minutes. In addition, the experiment results indicate its ability to detect both burst and low-and-slow exfiltrations.

Ghafir *et al.* [23] develop a system, called MLAPT, that detects APT based on network evidence. Unlike most systems that focus on one particular stage of an APT attack, MLAPT is able to detect various stages of an APT attack and predict the likelihood of an APT attack. MLAPT system can be categorized into three phases (*i.e.*, threat detection, alert correlation and attack prediction). Multiple modules that detect common techniques used in APT attacks are deployed to generate alerts and feed them to the correlation framework (FCI). The FCI framework then filters and clusters received alerts. In addition, the correlation between different clusters is evaluated. The final phase is a ML model that predicts the probability of a complete APT attack. The evaluation results demonstrate that MLAPT system achieves 81.8% true positive rate (TPR) and 4.5% false positive rate (FPR).

2.3.3 Host-Based APT Detection

Kaiafas *et al.* [32] successfully employ an ensemble of classifiers for detecting malicious events in the Los Alamos National Laboratory (LANL) dataset [40]. However, the authors are oblivious to the biased nature of the LANL dataset. Based on our analysis (*cf.*, Chapter 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 [32]) in training and inference. We highlight this limitation in Chapter 5.

Siadati *et al.* [56] propose APT-Hunter that visualizes the logons connection 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. It also achieves less than 50% recall on the LANL dataset. The authors also implement a system [57] that extracts logon patterns for anomaly detection. They propose a novel pattern mining algorithm that is scalable for large datasets. Their system 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 global financial institution is employed for evaluation. However, due to the lack of malicious activities, the authors inject attack traces according to penetration test campaigns. Their system yields 82% recall and 99.7% precision in detecting malicious logons. While the authors propose host-based anomaly detection that leverages pattern matching, the focus of our work is to harness ML techniques for anomaly detection.

Milajerdi *et al.* [50] 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 (HSG) 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, since it requires constant updates in order to detect new threats. Notably our system does not depend on any database to perform classification.

Ussath *et al.* [65] implement a user behavior simulation system to generate user activity logs for Windows platform. They leverage feed-forward neural networks and recurrent neural networks to identify malicious log events. However, the dataset generated by their simulation system is based on hypothetical assumptions. For example, the authors assume that all logon events pertaining to the start of user activities happen between 7:00 and 10:59 o'clock, which is unrealistic. In addition, some of their ML features, such as longitude and latitude of the user, are impractical for most real-world scenarios. For example, not all devices are equipped with GPS capabilities and IP-based geo-location may not be accurate due to VPN, to name a few.

Lopez and Sartipi [46] propose different feature extraction techniques and provide a list of features that can be employed for detecting Information System misuse. The authors demonstrate the usefulness of their proposed features using a simple logistic regression model, evaluated 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 possible features for addressing anomaly detection, they do not evaluate the performance of various ML techniques on the proposed features.

Creech *et al.* [14] designed 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 and their system yields 100% detection rate and 0.6% false alarm rate. While the approach in [14] 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 operating system architectures [15].

2.3.4 Hybrid APT Detection

Zeng *et al.* [70] build a botnet detection system that leverages both network and host information to make decisions. Their system consists of three major components (*i.e.*, host analyzer, network analyzer and correlation engine). The host analyzer is deployed on each host that monitors system-wide behaviors and uses SVM to generate a suspicion-level. While the network analyzer ingests flows from routers and applies a hierarchical clustering algorithm to group similarly-behaving hosts. The correlation engine collects information from the host analyzer if and only if a group of hosts are deemed to be suspicious by the network analyzer. Finally, the correlation engine assigns a detection score to each host and makes the final decision. Their system is evaluated on a semi-synthetic dataset, which consists of data collected from a campus network and a simulated environment. Their results indicate near zero false negative rate and acceptable FPR. Our approach is inspired from this work and several other works to develop two-staged classification of RDP sessions.

2.3.5 Adversarial Machine Learning

Biggio *et al.* [8] 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. Authors 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 approach. The proactive approaches can be categorized into security by design and security by obscurity, whereas, the reactive approach focuses on learning from the past. While both approaches can mitigate the risk of attacks, the authors suggest that their combination ensure a more secure system. This work helps us outline the different types of adversarial attacks that may compromise our proposed model.

Apruzzese *et al.* [2] study a network-based intrusion detection system [59] that uses ML techniques. The analyzed system uses network flow-based features with a random forest classifier for detecting botnet. The CTU [22] dataset is used for evaluating the detection system. 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 these results indicate that such a classifier is fragile to adversarial attacks, authors do not provide any solution that can mitigate this critical problem. While this study analyzes the weakness of a network-based intrusion detection system, this thesis develops a similar benchmark approach to prove that our system is robust against this type of adversarial attack.

Chapter 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 [22, 55], facilitate network-based intrusion detection. In contrast, host event logs contain sensitive information limiting their distribution by organizations [65]. To overcome this limitation, researchers (*e.g.*, [65]) 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* [40] and *unified* [64] datasets. In the combined dataset (*cf.*, Chapter 4), the Windows event IDs of interest to this thesis are 4624, 4625 and 4634, which pertain to RDP authentication. Table 3.1 provides a description of these events.

Table 3.1: Windows event ID reference [64]

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 [40] 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 thesis, 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 [64] is collected within LANL over a 90 day interval. Table 3.2 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 [31].

Table 3.2: 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	0x9c279eb	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	Descrition 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

Chapter 4

Methodology

4.1 Combining Datasets

Both datasets have limitations according to their authors [41] and our observations. Hence we decided to inject red team events from the comprehensive dataset [40] into the unified dataset [64]. 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 new dataset to avoid any bias in the ML classifier. However, we did not choose this approach to preserve the authenticity of the attacks.

Instead, we proceed as follows. 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. The distribution of benign and red team events in the first 30 days is illustrated in Figure 4.1.

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 */

$\mu \leftarrow Benign.sessions.duration.mean()$ ▷ Mean of session duration of *Benign*

$\sigma^2 \leftarrow Benign.sessions.duration.variance()$ ▷ Variance of session duration of *Benign*

$RedHosts \leftarrow Malicious.sourceHosts$ ▷ Set of source hosts in *Benign*

$BenignHosts \leftarrow Benign.sourceHosts$ ▷ Set of source hosts in *Malicious*

/* An authentication tuple is a combination of user name and destination host in an authentication event */

$RedTuples \leftarrow Malicious.authTuples$ ▷ Set of authentication tuples in *Benign*

$BenignTuples \leftarrow Benign.authTuples$ ▷ 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* */

$Source \leftarrow dict\{\}$

for each $host \in RedHosts$ **do**

    $Source[host] \leftarrow BenignHosts.randomPop()$

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* */

$AuthTuple \leftarrow dict\{\}$

for each $tuple \in RedTuples$ **do**

    $AuthTuple[tuple] \leftarrow BenignTuples.randomPop()$

End

/* Rewrite the fields in red team events and insert the modified event into *Benign* */

for each $event \in Malicious$ **do**

    $newSrc \leftarrow Source[event.SourceHost]$

    $newUser \leftarrow AuthTuple[event.AuthTuple].UserName$

    $newDst \leftarrow AuthTuple[event.AuthTuple].DestinationHost$

    $session \leftarrow GaussianRandom(\mu, \sigma^2)$

    $modified \leftarrow newEvent(event.timeStamp, newSrc, newUser, newDst, session)$

    $Benign.append(modified)$

End

return *Benign*

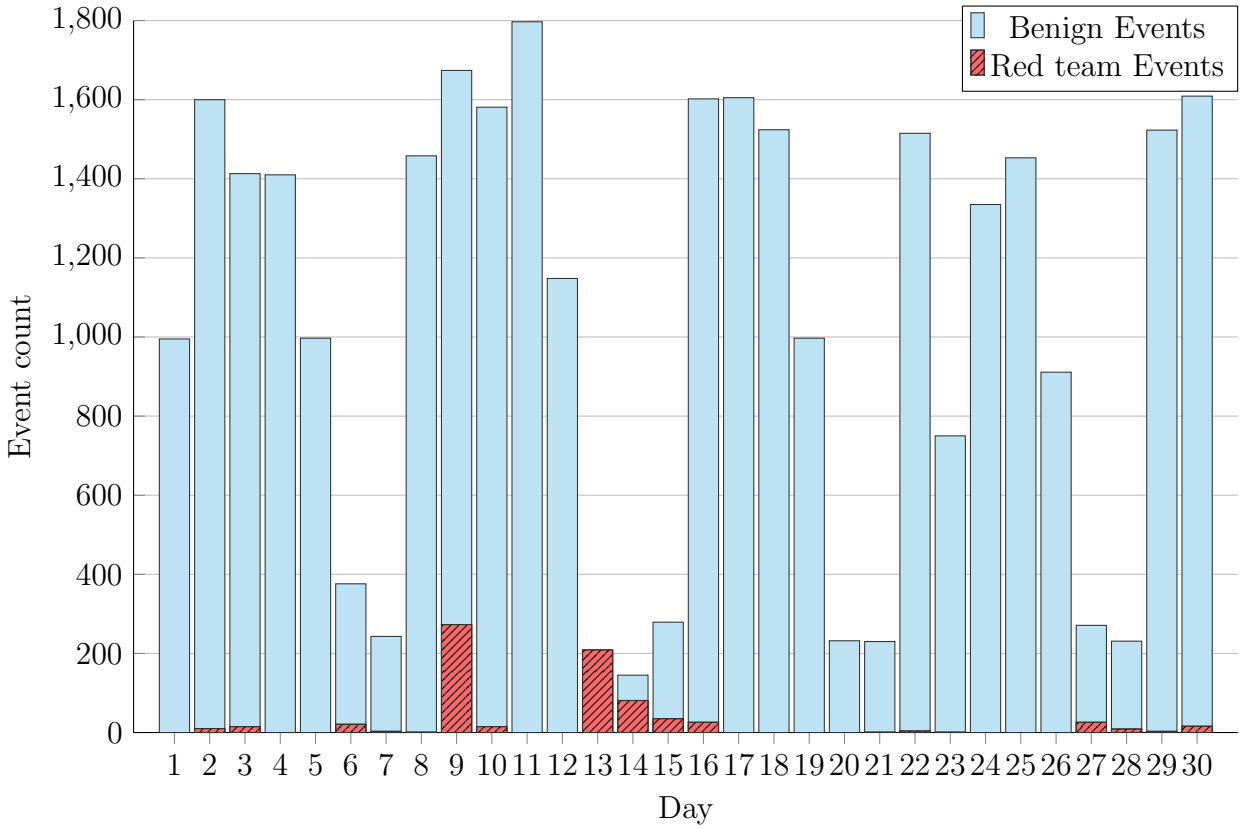

Figure 4.1: RDP events distribution

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.*, Chapter 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. It is important to 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.

4.2 Feature Engineering

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

- *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.

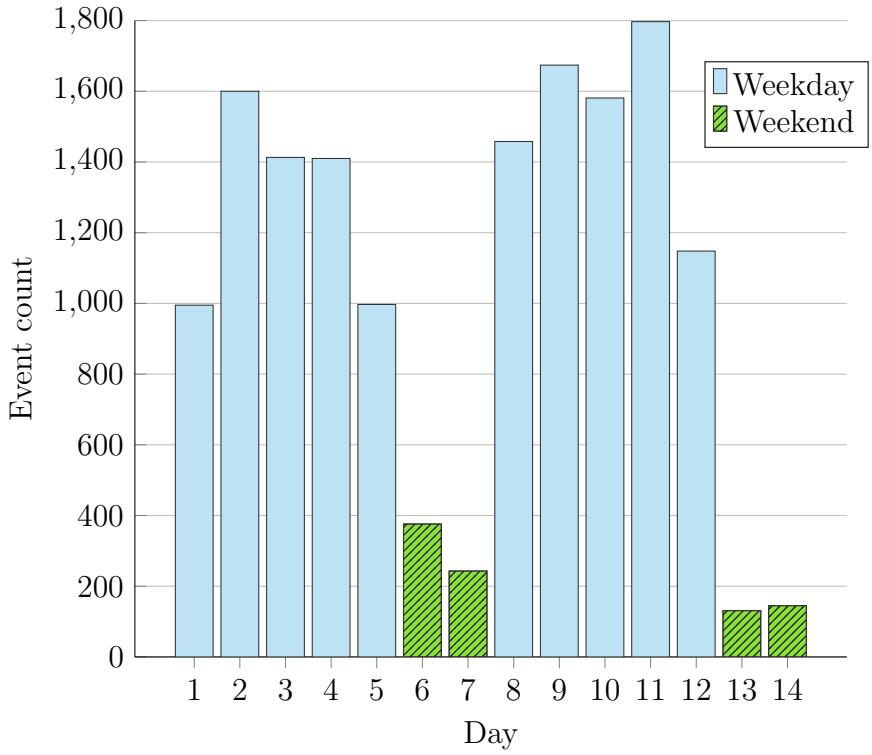

Figure 4.2: RDP events per day

Not all the attributes from the original dataset (*cf.*, Table 3.2) 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 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. 4.2. 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

We employ various ML techniques to evaluate our approach for detecting malicious RDP sessions. While the supervised ML algorithms exploit the relationship between the input data and its corresponding label, the unsupervised ML algorithms tend to learn the inherent structure of the input data without explicit labels.

4.3.1 Supervised Learning Algorithms

Based on previous studies [9,32,43,46,62,65], we select a variety of ML techniques that have proven effectiveness in anomaly 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 Decision Tree (DT) classifier with a maximum depth of three and criterion entropy. 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 (CART) algorithm [10]. Furthermore, we evaluate Random Forest (RF) [27], LogitBoost (LB) [21] and LightGBM [39], 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. LightGBM is similar to LB, and is a recent DT-based gradient boost algorithm. In comparison to other Gradient Boosting Decision Tree (GBDT), the efficiency and scalability of LightGBM is better by one order of magnitude [39]. We also evaluate Feed-forward Neural Network (FNN), a simple neural network without cycles between each layer.

4.3.2 Unsupervised Learning Algorithms

K-means [47] is one of the most popular clustering algorithms that first assigns data points to clusters based on existing centriods and then selects new centriods based on cluster assignment. It repeats these two steps until the assignments no longer update. Since the vanilla version of K-means does not scale with a large amount of data, we leverage Mini Batch K-means for reducing computation time. Agglomerative Clustering (AC) is another clustering approach that uses bottom-up hierarchical clustering. In agglomerative clustering, each observation starts as a singleton cluster, and the algorithm progressively merges pairs of clusters until they reduce to the desired number of clusters. The euclidean distance is

used as the distance metric in our model. Balanced Iterative Reducing and Clustering using Hierarchies (Birch) [71] is another hierarchical clustering approach. The Birch algorithm is able to incrementally produce clusters by ingesting new data points, which is useful for online learning. However, Birch does not scale well with higher dimension of feature space. In our experiment, we set its branching factor to 50 and threshold to 1000. We also employ Density-based Spatial Clustering of Applications with Noise (DBSCAN) [20], another commonly used clustering algorithm. The DBSCAN algorithm not only groups close data points into clusters but also mark points in low density area as outliers. Thus, it can potentially identify anomaly in our dataset. In our experiment, we set epsilon (EPS) to 3,000 and minimum samples to 10.

4.3.3 Metrics

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

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

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

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

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

$$AP\ score = \sum_n (\text{Recall}_n\text{-Recall}_{n-1}) \times \text{Precision}_n$$

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 score 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 Receiver Operating Characteristic (ROC) curve can illustrate the aggregate performance of a classifier, it suffers with imbalanced dataset as well. Hence, we do not include it in our metrics.

Chapter 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 3.30GHz CPU and 16GB RAM. Nodes are interconnected with 10Gbps Ethernet. Model training and validation for supervised learning approaches are performed on an Amazon AWS EC2 t3.medium instance. Due to the hardware resource limitation of AWS instance, the unsupervised learning phase is conducted on a machine that has 2 x Intel(R) Xeon(R) Silver 4114 CPU @ 2.20GHz CPU and 196GB RAM.

5.1.2 Software

A Logstash instance is deployed to ingest the dataset into an ElasticSearch [24] cluster, and Kibana is used for data visualization. For data pre-processing, a variety of Python packages, including Numpy [52], Scipy [30] and Pandas [49] are employed. The ML models are developed in Python with Scikit-learn [53] and Keras [13] libraries.

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 5.1. 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.*, Chapter 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 [4]. Hence, we do not explore sampling techniques in this thesis. 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 [21]. Another boosting algorithm, LightGBM, achieves a slightly poorer performance than LB in precision and recall. Even though the probabilistic GNB classifier under performs the DT family of classifiers, it outperforms LR and FNN.

Table 5.1: RDP session detection with all baseline features

Classifier	Accuracy	Precision	Recall	F_1
Logistic Regression	98.50%	10.93%	1.74%	0.030
Decision Tree	99.90%	99.04%	93.58%	0.962
Feed-forward NN	98.68%	0%	0%	0
Gaussian-NB	99.60%	87.31%	82.11%	0.846
Random Forest	99.95%	99.73%	96.13%	0.979
LogitBoost	99.99%	99.87%	99.73%	0.998
LightGBM	99.99%	99.73%	99.33%	0.995

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 is a misfit 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 as to run a *robustness* test. As shown in Table 5.2, this results in over-fitting, with RF unable to correctly classify any malicious RDP session from unknown source hosts.

Table 5.2: Robustness of standalone 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

Therefore, we remove from our feature set those features that cause such a bias, namely username, source host and destination host. Table 5.3 depicts the result after the removal of these features. In comparison to Table 5.1, no significant difference is evident in terms of precision, recall or F_1 score in classifying RDP sessions. The LightGBM classifier achieves perfect precision in this test case even though the overall performance decreased 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 5.3: RDP session classification (*user*, *src* and *dst* features removed)

Classifier	Accuracy	Precision	Recall	F_1
Logistic Regression	98.50%	11.34%	1.87%	0.321
Decision Tree	99.90%	99.04%	93.58%	0.962
Feed-forward NN	98.68%	0%	0%	0
Gaussian-NB	99.60%	87.31%	82.11%	0.846
Random Forest	99.94%	99.59%	96.13%	0.978
LogitBoost	99.99%	99.87%	99.47%	0.997
LightGBM	99.99%	100%	98.8%	0.994

5.2.1 Two-Stage Classification

There are a few works that employ multiple stage training for improving the performance of classification. For example, Kim *et al.* [43] propose a hierarchical approach that decomposes

Table 5.4: Two-stage classification with (*user*, *src*, and *dst* features removed)

Classifiers	Accuracy	Precision	Recall	F_1	Training Time (s)
K-means+LB	99.99%	99.87%	99.47%	0.997	10.21
AC+LB	99.99%	99.87%	99.47%	0.997	40.63
Birch+LB	99.99%	99.74%	99.33%	0.995	24.64
DBSCAN+LB	99.99%	99.87%	99.47%	0.997	10.77
LB	99.99%	99.87%	99.47%	0.998	9.26

normal training data into smaller subsets using DT and leverage one-class support vector machine (SVM) for each subset. Chitrakar *et al.* [12] propose a similar approach, where the training data is split into different clusters using k-medoids, followed by naïve bayes for further attack classification. Therefore, we set out to boost the performance of our stand-alone classifiers using two-stage classification. We first employ unsupervised learning algorithm to group RDP sessions into clusters and treat the cluster information (which cluster a datapoint belongs to) as a new input feature. After that, the RDP sessions are fed through supervised learning for further classification. The rationale for this approach is that benign RDP sessions may form clusters according to a particular user's behavior. It would be helpful for classification algorithm to make a decision with additional clustering information that leverages user behavior.

In this experiment, we select LB for the classification stage since it has the best aggregate performance. The group of clustering algorithms employed in conjunction with LB is discussed in Chapter 4. The number of cluster is set to 100 for all clustering algorithms except DBSCAN (that does not accept number of clusters as input parameter) and the result is depicted in Table 5.4. Evidently, no improvements can be observed over the stand-alone LB classifier with an increase in training time. In order to further investigate the potential of unsupervised learning algorithms, the subsequent experiment focuses on a single algorithm with different number of clusters. The Birch and AC algorithms are removed from the candidate algorithms as they require too much time for computing clusters. The DBSCAN is not under consideration as well, since it is not straightforward to control the number of clusters. Specially, the number of clusters are calculated automatically based on other hyper-parameters, such as eps. Therefore, the only candidate left is K-means and the corresponding results are illustrated in Fig. 5.1. The training time in the figure includes both clustering and classification stages. Notably, the performance of two-stage classification surpasses the stand-alone LB in terms of recall when the number of clusters equals to 1000. However, this improvement is marginal. Furthermore, the training time of K-means grows exponentially with the data size [69], making it infeasible in practice.

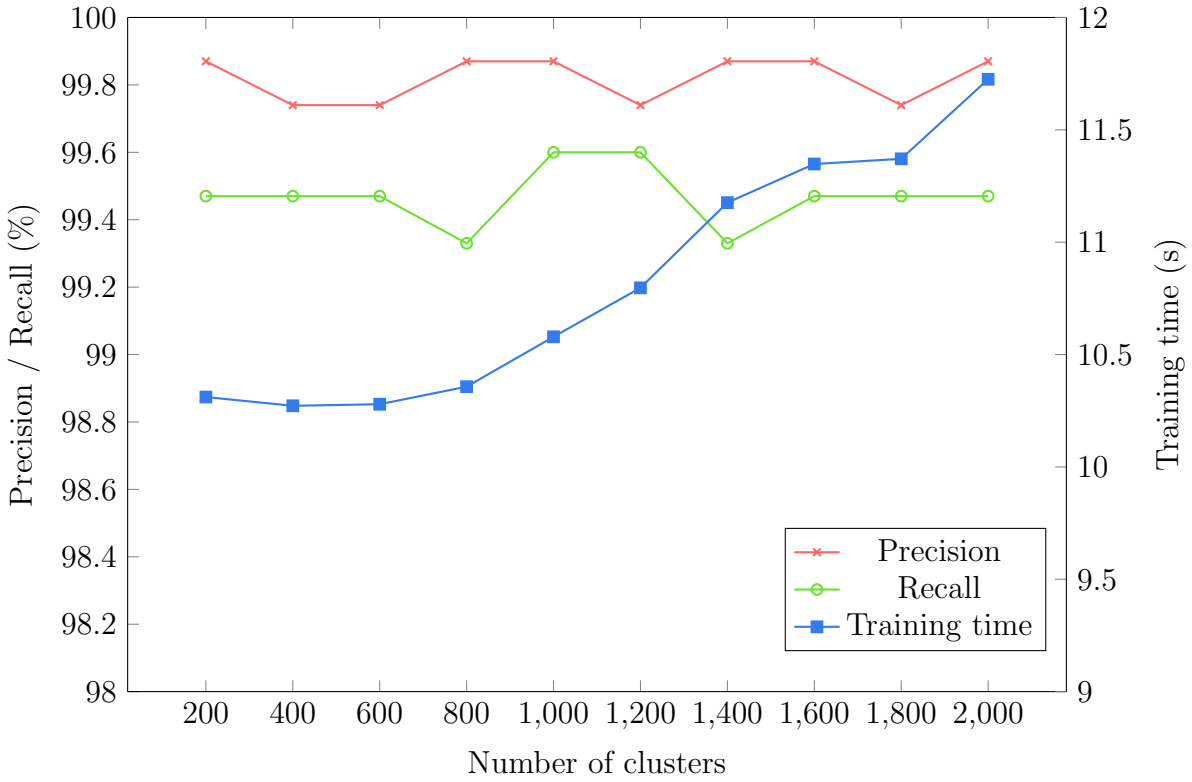

Figure 5.1: Recall, precision and training time *vs.* # of clusters for K-means with LB

5.2.2 Voting

The authors in [32] 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 5.5. 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 5.3), 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 LightGBM is removed since LogitBoost has better performance). The performance of the combined classifiers is shown in Table 5.6. In comparison to the previous ensemble depicted in Table 5.5, 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.

Table 5.5: 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 5.6: 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

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 select the best performing ensemble of classifiers from Table 5.6. The first three columns in Table 5.7 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 the weights of 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 5.7. 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 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. 5.2, while precision and

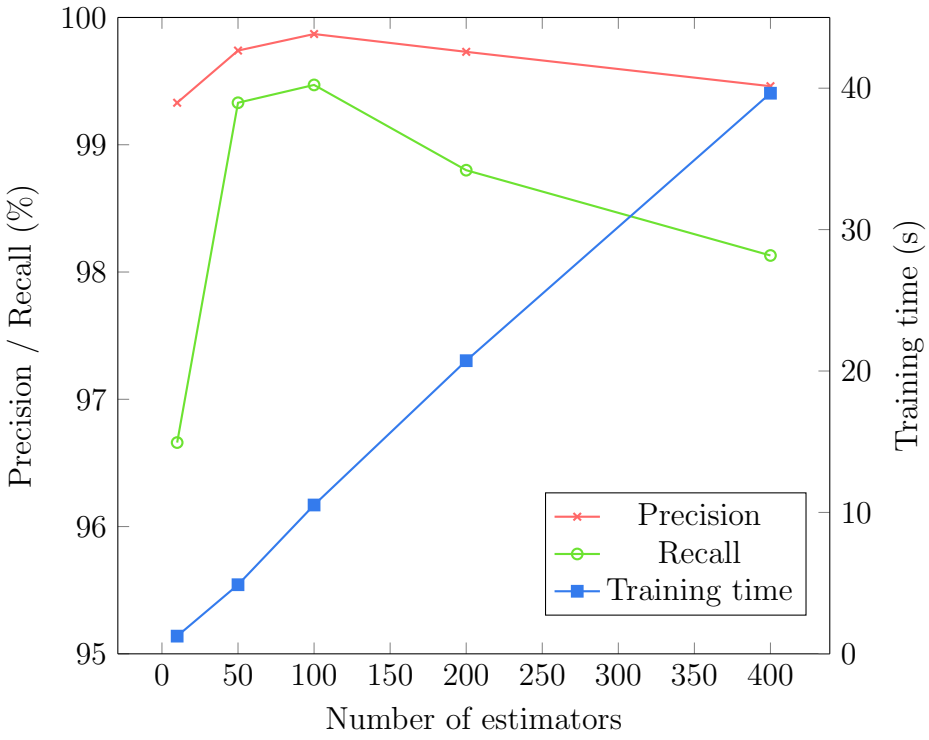

Figure 5.2: Recall, precision and training time *vs.* # of estimators for stand-alone LB (our model)

recall are the highest with around 100 estimators. In the remaining experiments, we will use stand-alone LB with 100 estimators.

Table 5.7: Weighted voting for RDP session classification using LB, RF and GNB classifiers (*user*, *src*, and *dst* features removed)

LB	RF	GNB	Threshold	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

5.2.3 Comparative Analysis

We compare our stand-alone LB classifier with Kaiafas *et al.* [32]. The LB classifier is preferred over the LightGBM because we do not want to miss any attack. Although LightGBM achieves perfect precision in our previous experiment (*cf.*, Table 5.3), its recall is lower than LB. As mentioned in Chapter 1, our goal in this thesis 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.

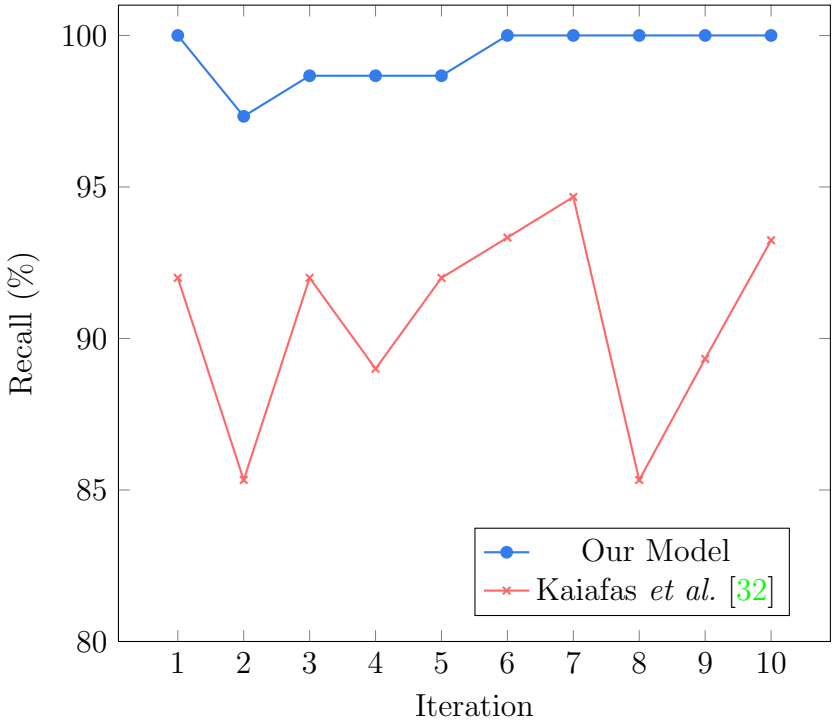

Figure 5.3: Recall during each iteration

In order to compare, we implement Kaiafas's [32] approach and evaluate the corresponding model on our dataset. A list of the features used in their work is available in Appendix A. 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 5.8, with all available features, the recall of Kaiafas's model is slightly lower than our model (first two rows without *). Though 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's model has a significant drop in

recall from 98.67% to 90.66% (last two rows with *). In addition, the standard deviation of recall is high for [32]. For some iterations of cross validation, it can only achieve 85% recall, as shown in Fig. 5.3. On the other hand, our model illustrates stability in recall over multiple iterations. Furthermore, the training time of Kaiafas's 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 the state-of-the-art in RDP session classification in terms of both performance and training time.

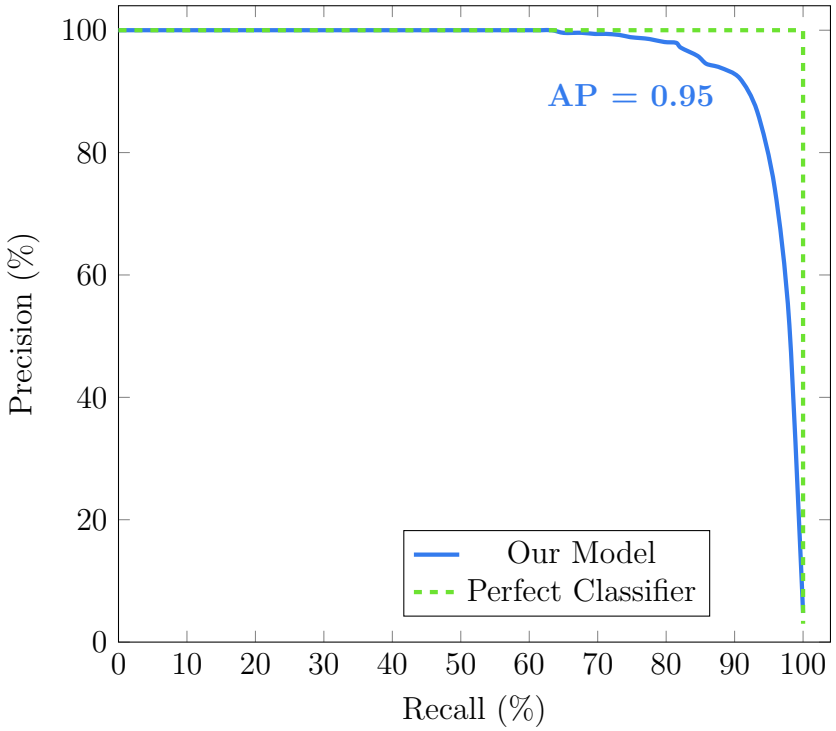

Figure 5.4: Precision-Recall curve of our model

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 contain attacks originating from three different sources, the testing set contains an additional attacking source that does not appear in the training set. In Fig. 5.4, 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,

Table 5.8: RDP session classification using stand-alone LB vs. [32]

Classifier	Accuracy	Precision	Recall	F_1	Training Time (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
*Our Model	99.98%	99.87%	99.47%	0.992	10.53
*Kaiafas *et al.*	99.88%	100.00%	90.66%	0.951	18.19

* = Model validation without *user*, *src* and *dst* features

such as Kaiafas's 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 obtains the highest F_1 score of 0.914, Kaiafas's model scores as low as 0.675.

5.2.4 Robustness to Adversarial Attempts

ML algorithms were originally designed without considering attackers that may intentionally fabricate the input data to manipulate the outcome of a classifier [6, 28]. These algorithms assume a benign environment, where both training and testing datasets are stationary, and follow the same statistical distribution. According to [5], these adversarial attacks can be categorized into different taxonomies along three axes (*i.e.*, attack influence, attack specificity and security violation). A brief summary of these axes are presented in Table 5.9. In addition, we list the examples of potential adversarial attacks against our classification model in Table 5.10 based on the previous summary in Table 5.9. This table helps us focus our study on a plausible adversarial scenario.

In this thesis, we focus on exploratory attacks with the assumption that attackers do not have access to our classifier and training dataset. Since our model heavily depends on the benign user behavior, once the adversaries obtain access to the training dataset, they can easily mimic a benign user's logon pattern by learning their authentication and communication patterns. Due to this limitation, our model will perform poorly under such circumstances. In addition, we do not consider causative attacks in this thesis. Since we assume that the attackers have no direct access to our training dataset (or the classifier itself), the only way they can mislead our classifier is to generate massive RDP sessions within a protected network. As a result, our classifier will become biased during the re-training phase. However, this hypothetical scenario is unrealistic, since system administrators will be alerted by our model of such a large invasion. They can investigate the root cause and

patch any vulnerability.

Table 5.9: Taxonomy of attacks against ML systems [6, 28]

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 5.10: Examples of potentials 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

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. 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 by percentages. An example of an adversarial sample is shown in Table 5.11.

The testing dataset in subsequent experiments only contain malicious RDP sessions. Thus, the precision is always 100%, while the accuracy is always equal to recall. Therefore,

Table 5.11: Example of polymorphic form of an attack

Data	Weekday	Seconds in a day	Session duration	User time diff	Source time diff	Destination time diff
Original	2	12340	100	1000	1000	1000
Mutation	6 (random)	45670 (random)	125 (+25%)	1250 (+25%)	1250 (+25%)	1250 (+25%)

we use accuracy as the evaluation metric in the these experiments. In the first experiment, we train our model on the entire dataset from Chapter 4 and created 300 new malicious RDP sessions for testing, by employing the aforementioned perturbations. Specifically, we randomly select 300 malicious data points from the training set and mutate them. The classification result of newly crafted RDP sessions is illustrated in Fig. 5.5. The positive change implies that the original feature values are increased by certain percentages, while, the negative change means that the original feature values are decreased by certain percentages. In Fig. 5.5, 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. 5.6. 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. 5.6 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.

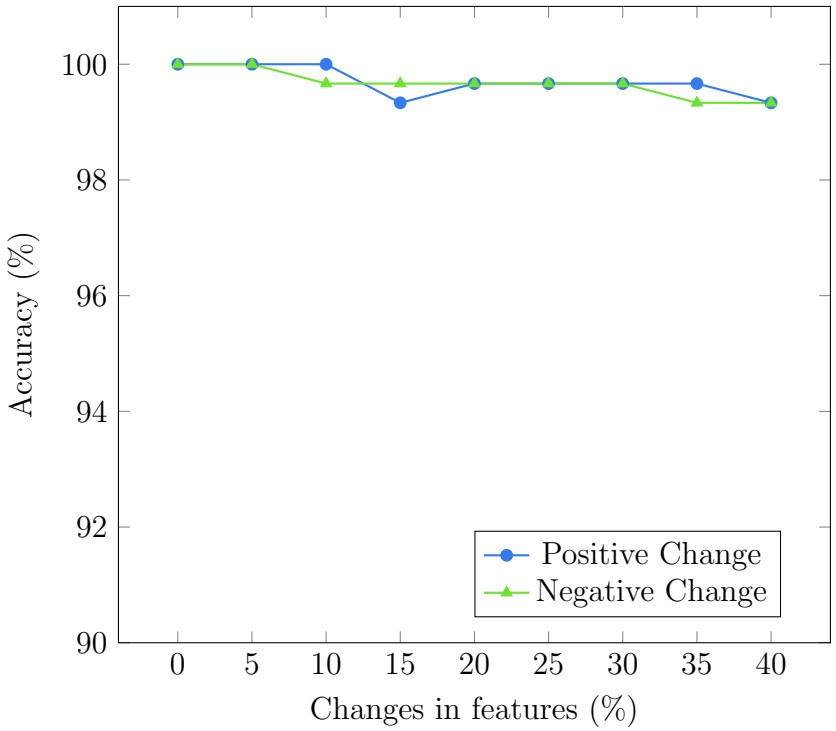

Figure 5.5: Detection accuracy for polymorphic forms of *known* attack

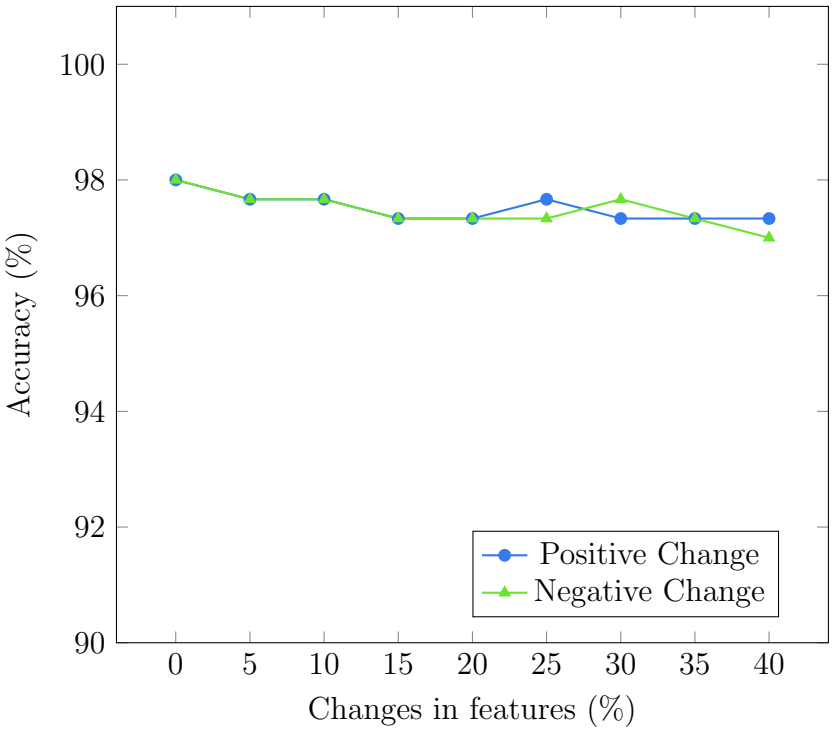

Figure 5.6: Detection accuracy for polymorphic forms of *unknown* attack

Chapter 6

Conclusion and Future Works

6.1 Conclusion

The APT is one of the most sophisticated and evolving cyber attacks and it has gained a lot of attention in the past decade. APT not only persistently collects data from a compromised target, but also leverages the existing victim to progressively spread throughout the network by exploiting various system vulnerabilities. In this thesis, we study existing systems and approaches for anomaly detection via host events. Since RDP is one of the major tools employed during lateral movement stage, we leverage Windows event logs for anomaly-based 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 extract relevant features, and explore both supervised and unsupervised learning algorithms to detect anomalous RDP sessions. After evaluating various clustering and classification algorithms, we chose stand-alone LB as the best model with respect to accuracy, recall and precision in classifying RDP sessions. LB shows promising results and outperforms a state-of-the-art model [32] in recall and training time for detecting malicious Windows RDP sessions. In addition, we demonstrate that our approach is robust to adversarial attacks.

6.2 Recommendations

The dataset has always been a significant factor that influences the results of a ML study. However, it is difficult to find an ideal dataset in some cases. We would make a few

recommendations regarding data collection. The first approach is to search for datasets that are publicly available on the internet. The second approach is to set up the infrastructure and capture data if no available dataset can be found. In the case which human interaction is required, it is possible to recruit voluntary participants by providing some rewards. In the worst case scenario, the user's behavior can be simulated to generate the synthetic dataset.

6.3 Future Works

There are a few extensions of this work that can be explored as future research directions:

Online Learning Due to the large amount of log data generated daily in organizations, their storage not only increases operating costs but also wastes hardware resources. Furthermore, training ML models from scratch can be computationally intensive, time consuming, and prohibitive. Therefore, it is crucial to retrain ML models as new data becomes available, thus accommodating ML models boundary changes after deployment.

Hybrid System A hybrid anomaly detection system that leverages data from different sources may be beneficial. There are few works [11,70] that attempt to build a system/model that correlates network flows and host logs. A combination of host-based and network-based APT detection system can build an advanced detection system that attackers cannot easily evade. It is feasible to combine our approach with another network-based anomaly detection system to further boost detection rate.

Other Session-based Protocol The model presented in this thesis for RDP can work with other session-based protocols. For example, we can evaluate our model on SSH sessions with datasets in the future.

More Use Cases of Event Logs The Windows event log contains a variety of event types, which can be leveraged to identify different stages of an APT attack. However, our current work is only limited to one specific tool used in the LM stage. It is possible to leverage system events other than authentication to classify between benign and unauthorized use of system administration tools. In addition, the execution of malicious tools or malware can be identified by analyzing event logs as well.

References

[1] Remote desktop protocol. https://docs.microsoft.com/en-us/windows/desktop/termserv/remote-desktop-protocol. Accessed: 2019-02-22.

[2] Giovanni Apruzzese and Michele 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)*, pages 1–8. IEEE, 2018.

[3] Sara Ayoubi, Noura Limam, Mohammad A Salahuddin, Nashid Shahriar, Raouf Boutaba, Felipe Estrada-Solano, and Oscar M Caicedo. Machine learning for cognitive network management. *IEEE Communications Magazine*, 56(1):158–165, 2018.

[4] Pierre Baldi. Autoencoders, unsupervised learning and deep architectures. In *Proceedings of the International Conference on Unsupervised and Transfer Learning Workshop*, pages 37–50, 2011.

[5] Marco Barreno, Blaine Nelson, Anthony D Joseph, and J Doug Tygar. The security of machine learning. *Machine Learning*, 81(2):121–148, 2010.

[6] Marco Barreno, Blaine Nelson, Russell Sears, Anthony D Joseph, and J Doug Tygar. Can machine learning be secure? In *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, pages 16–25. ACM, 2006.

[7] Konstantin Berlin, David Slater, and Joshua Saxe. Malicious behavior detection using windows audit logs. In *Proceedings of the ACM Workshop on Artificial Intelligence and Security (AISec)*, 2015.

[8] Battista Biggio, Giorgio Fumera, and Fabio Roli. Pattern recognition systems under attack: Design issues and research challenges. *International Journal of Pattern Recognition and Artificial Intelligence*, 28(07):1460002, 2014.

[9] Raouf Boutaba, Mohammad A Salahuddin, Noura Limam, Sara Ayoubi, Nashid Shahriar, Felipe Estrada-Solano, and Oscar M Caicedo. A comprehensive survey on machine learning for networking: evolution, applications and research opportunities. *Journal of Internet Services and Applications*, 9(1), 2018.

[10] Leo Breiman, Jerome H Friedman, Richard A Olshen, and Charles J Stone. *Classification and regression trees*. The Wadsworth statistics/probability series. Wadsworth & Brooks/Cole Advanced Books & Software, Monterey, CA, 1984.

[11] 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*, pages 708–715, 2018.

[12] Roshan Chitrakar and Chuanhe 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*, pages 1–5, 2012.

[13] François Chollet et al. Keras. https://keras.io, 2015.

[14] G. Creech and J. Hu. A semantic approach to host-based intrusion detection systems using contiguousand discontiguous system call patterns. *IEEE Transactions on Computers*, 63(4), April 2014.

[15] Gideon Creech. *Developing a high-accuracy cross platform Host-Based Intrusion Detection System capable of reliably detecting zero-day attacks.* PhD thesis, University of New South Wales, Canberra, Australia, 2014.

[16] Crowdstrike. Deep in thought: Chinese targeting of national security think tanks, 2014. Accessed: 2019-08-08.

[17] Cylance. Operation cleaver, 2014. Accessed: 2019-08-08.

[18] Abbas Abou Daya, Mohammad A. Salahuddin, Noura Limam, and Raouf Boutaba. A graph-based machine learning approach for bot detection, Feb 2019.

[19] Jon DiMaggio. The black vine cyberespionage group, Aug 2015. Accessed: 2019-02-28.

[20] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[21] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28:2000, 1998.

[22] S. García, M. Grill, J. Stiborek, and A. Zunino. An empirical comparison of botnet detection methods. *Comput. Secur.*, 45, 2014.

[23] Ibrahim Ghafir, Mohammad Hammoudeh, Vaclav Prenosil, Liangxiu Han, Robert Hegarty, Khaled Rabie, and Francisco J Aparicio-Navarro. Detection of advanced persistent threat using machine-learning correlation analysis. *Future Generation Computer Systems*, 89:349–359, 2018.

[24] Clinton Gormley and Zachary Tong. *Elasticsearch: The Definitive Guide*. O'Reilly Media, Inc., 1st edition, 2015.

[25] Fox-IT Group-IB. Anunak:apt against financial institutions, 2014. Accessed: 2019-08-08.

[26] Thoufique Haq, Jinjian Zhai, and Vinay K Pidathala. Advanced persistent threat (apt) detection center, April 18 2017. US Patent 9,628,507.

[27] Tin Kam Ho. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE, 1995.

[28] Ling Huang, Anthony D Joseph, Blaine Nelson, Benjamin IP Rubinstein, and JD Tygar. Adversarial machine learning. In *Proceedings of the 4th ACM workshop on Security and artificial intelligence*, pages 43–58. ACM, 2011.

[29] Eric M Hutchins, Michael J Cloppert, and Rohan M Amin. Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. *Leading Issues in Information Warfare & Security Research*, 1, 2011.

[30] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. Accessed Mar 2019.

[31] JPCERT Coordination Center. Detecting lateral movement through tracking event logs, Dec 2017. Accessed: 2019-02-26.

[32] G. Kaiafas, G. Varisteas, S. Lagraa, R. State, C. D. Nguyen, T. Ries, and M. Ourdane. Detecting malicious authentication events trustfully. In *Proceedings of NOMS*, April 2018.

[33] Stamatis Karnouskos. Stuxnet worm impact on industrial cyber-physical system security. In *Proceedings of IECON Annual Conference of the IEEE Industrial Electronics Society*, 2011.

[34] Kaspersky Lab. The icefog apt: A tale of cloak and three daggers, 2013. Accessed: 2019-08-08.

[35] Kaspersky Lab. The regin platform nation-state ownage of gsm networks, 2014. Accessed: 2019-08-08.

[36] Kaspersky Lab. Carbanak apt: The great bank robbery, 2015. Accessed: 2019-03-03.

[37] Kaspersky Lab. The duqu 2.0, 2015. Accessed: 2019-08-08.

[38] Kaspersky Lab. The msnmm campaigns, 2015. Accessed: 2019-08-08.

[39] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, pages 3146–3154, 2017.

[40] Alexander D. Kent. Comprehensive, Multi-Source Cyber-Security Events. Los Alamos National Laboratory, 2015.

[41] Alexander D. Kent. Proceedings of Cybersecurity Data Sources for Dynamic Network Research. In *Dynamic Networks in Cybersecurity*, June 2015.

[42] S. Khattak, N. R. Ramay, K. R. Khan, A. A. Syed, and S. A. Khayam. A taxonomy of botnet behavior, detection, and defense. *IEEE Communications Surveys Tutorials*, 16(2), Second 2014.

[43] Gisung Kim, Seungmin Lee, and Sehun Kim. A novel hybrid intrusion detection method integrating anomaly detection with misuse detection. *Expert Systems with Applications*, 41(4):1690–1700, 2014.

[44] Brian Krebs. Anthem breach may have started in april 2014. https://krebsonsecurity.com/2015/02/anthem-breach-may-have-started-in-april-2014/, 2015. Accessed: 2019-02-22.

[45] Hung-Jen Liao, Chun-Hung Richard Lin, Ying-Chih Lin, and Kuang-Yuan Tung. Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, 36(1):16 – 24, 2013.

[46] Eduardo Lopez and Kamran 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.

[47] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[48] Mirco Marchetti, Fabio Pierazzi, Michele Colajanni, and Alessandro Guido. Analysis of high volumes of network traffic for advanced persistent threat detection. *Computer Networks*, 109:127 – 141, 2016. Traffic and Performance in the Big Data Era.

[49] Wes McKinney. Data structures for statistical computing in python. In *Proceedings of the Python in Science Conference*, 2010.

[50] S. Momeni Milajerdi, R. Gjomemo, B. Eshete, R. Sekar, and V. Venkatakrishnan. Holmes: Real-time apt detection through correlation of suspicious information flows. In *2019 2019 IEEE Symposium on Security and Privacy (SP)*, pages 447–462, Los Alamitos, CA, USA, may 2019. IEEE Computer Society.

[51] Daesung Moon, Sung Bum Pan, and Ikkyun Kim. Host-based intrusion detection system for secure human-centric computing. *The Journal of Supercomputing*, 72(7), 2016.

[52] Travis E. Oliphant. *Guide to NumPy*. CreateSpace Independent Publishing Platform, USA, 2nd edition, 2015.

[53] Pedregosa et al. Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.*, 12, November 2011.

[54] Dell EMC RSA. Rsa incident response: Emerging threat profile shell_crew, 2014. Accessed: 2019-08-08.

[55] Elaheh Biglar Beigi Samani, Hossein Hadian Jazi, Natalia Stakhanova, and Ali A. Ghorbani. Towards effective feature selection in machine learning-based botnet detection approaches. *IEEE Conference on Communications and Network Security*, pages 247–255, 2014.

[56] H. Siadati, B. Saket, and N. Memon. Detecting malicious logins in enterprise networks using visualization. In *2016 IEEE Symposium on Visualization for Cyber Security (VizSec)*, pages 1–8, Oct 2016.

[57] Hossein Siadati and Nasir Memon. Detecting structurally anomalous logins within enterprise networks. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 1273–1284, 2017.

[58] Sana Siddiqui, Muhammad Salman Khan, Ken Ferens, and Witold 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.

[59] Matija Stevanovic and Jens Myrup Pedersen. An analysis of network traffic classification for botnet detection. In *2015 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA)*, pages 1–8. IEEE, 2015.

[60] Colin Tankard. Advanced persistent threats and how to monitor and deter them. *Network Security*, 2011(8):16 – 19, 2011.

[61] FortiGuard SE Team. As the holiday season draws near, mobile malware attacks are prevalent, Nov 2018.

[62] Chih-Fong Tsai, Yu-Feng Hsu, Chia-Ying Lin, and Wei-Yang Lin. Intrusion detection by machine learning: A review. *Expert Systems with Applications*, 36(10), 2009.

[63] Yu Tsuda, Junji Nakazato, Yaichiro Takagi, Daisuke Inoue, Koji Nakao, and Kenjiro Terada. A lightweight host-based intrusion detection based on process generation patterns. In *Proceedings of Asia Joint Conference on Information Security (AsiaJCIS)*, 2018.

[64] Melissa J. M. Turcotte, Alexander D. Kent, and Curtis Hash. *Unified Host and Network Data Set*, chapter Chapter 1. World Scientific, Nov 2018.

[65] M. Ussath, D. Jaeger, F. Cheng, and C. Meinel. Identifying suspicious user behavior with neural networks. In *Proceeding of IEEE Intl. Conf. on Cyber Security and Cloud Computing*, June 2017.

[66] Martin Ussath, David Jaeger, Feng Cheng, and Christoph Meinel. Advanced persistent threats: Behind the scenes. In *Proceedings of Annual Conference on Information Science and Systems (CISS)*, 2016.

[67] John R Vacca. *Network and system security*. Elsevier, 2013.

[68] Andrew 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.

[69] Andrea Vattani. K-means requires exponentially many iterations even in the plane. *Discrete & Computational Geometry*, 45(4):596–616, 2011.

[70] Y. Zeng, X. Hu, and K. G. Shin. Detection of botnets using combined host- and network-level information. In *2010 IEEE/IFIP International Conference on Dependable Systems Networks (DSN)*, pages 291–300, June 2010.

[71] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: an efficient data clustering method for very large databases. In *ACM Sigmod Record*, volume 25, pages 103–114. ACM, 1996.

APPENDICES

Appendix A

Features Used by Kaiafas's Work

features	**explanation**
Median	Median of time difference of events between systems and from user to system
95th percentile	95th percentile of time difference of events between systems and from user to system
Standard Deviation	Standard deviation of time difference of events between systems and from user to system
Frequency	The amount of past similar events
First Occurrence	A flag denoting an event without any prior similar event
Geometric Distribution	Distribution of malicious events within a sequence of similar events
Popular User	User of the most occurrences within a sequence of similar events
Diversity	Number of different users within a sequence of similar events