Detecting Structurally Anomalous Logins Within Enterprise Networks

Hossein Siadati
New York University
Brooklyn, NY
hossein@nyu.edu

Nasir Memon
New York University
Brooklyn, NY
memon@nyu.edu

ABSTRACT

Many network intrusion detection systems use byte sequences to detect lateral movements that exploit remote vulnerabilities. Attackers bypass such detection by stealing valid credentials and using them to transmit from one computer to another without creating abnormal network traffic. We call this method Credential-based Lateral Movement. To detect this type of lateral movement, we develop the concept of a *Network Login Structure* that specifies normal logins within a given network. Our method models a network login structure by automatically extracting a collection of login patterns by using a variation of the market-basket analysis algorithm. We then employ an anomaly detection approach to detect malicious logins that are inconsistent with the enterprise network's login structure. Evaluations show that the proposed method is able to detect malicious logins in a real setting. In a simulated attack, our system was able to detect 82% of malicious logins, with a 0.3% false positive rate. We used a real dataset of millions of logins over the course of five months within a global financial company for evaluation of this work.

CCS CONCEPTS

• **Security and privacy → Intrusion/anomaly detection and malware mitigation**; **Network security**; • **Computing methodologies → Anomaly detection**;

KEYWORDS

Network Lateral Movement, Anomalous Logins, Pattern Mining

1 INTRODUCTION

Enterprise networks have been frequent targets of data breaches and sabotage [5, 16, 29, 34]. A common theme of these attacks is to follow a step by step process of chained computer hacking to reach a planned target computer. In these attacks, the perpetrator steals and uses credentials to compromise the next computer in the chain. Such an attacker begins by setting up a foothold in a network by compromising one computer, often by spear phishing. The attacker then steals passwords of network users and uses them to log in to other computers. Doing this, the attacker moves laterally between computers until obtaining access to critical data located deeper inside the network. We call this method of attack Credential-based Lateral Movement (CLM). Attackers have used this technique in many instances of data breaches, including the JP Morgan Chase [29] and Target hacks [16].

Traditional network intrusion detection systems (NIDS) detect malicious network traffic that signifies execution of a remote exploit. Since the content of network traffic in CLM is indistinguishable from a benign login, NIDS is not useful in the detection of CLM. On the other hand, access control policies and tools (e.g., Access Control Lists, Active Directory) fail to minimize the possible paths of lateral movement, due to obstacles faced in an enterprise environment in achieving a perfect implementation of the *principle of least privilege* [26]. Access control is usually relaxed to facilitate business continuity and to enable recovery of computer services when they fail. Therefore, permissions are provisioned for the worst case scenarios, allowing logins that would not usually be required. Sinclair et al. [30] have studied the problem of access controls in enterprise networks and showed that 50-90% of users are over-entitled regarding what they can access. This situation allows attackers to use stolen credentials to roam easily within a network and capture their target destinations.

In this paper, we present a method for detection of an important subclass of malicious logins within enterprise networks. Our method relies on two observations. First, the login connectivities of users within an enterprise are structured and mostly predictable. For example, staff of the human resource department connect to a server hosting an HR application, but employees of the accounting department connect to a server hosting an accounting application. Second, CLMs often involve connections between computers, that are not consistent with the login structure of an enterprise network. For example, an attacker might use a stolen credential to log in from a computer in the HR department to a computer in the accounting department, which is not a typical destination for computers of the HR department. These inconsistencies are inevitable as the attacker can only use stolen credentials he has and computers that he has already compromised to move forward. The difficulty in detecting such unusual movements is arriving at a characterization of normal login patterns in a complex enterprise system, and detecting abnormal logins without incurring high false positives that are inevitable due to the base rate fallacy [2]. Hence, we develop the concept of *Network Login Structure* that specifies normal logins within a given network. We develop a method to model a network login structure by automatically extracting a collection of login patterns. These patterns describe how groups of users typically log in between a group of computers. We then employ an anomaly

ACM ISBN ISBN 978-1-4503-4946-8/17/10…$15.00
https://doi.org/http://dx.doi.org/10.1145/3133956.3134003

detection approach to detect malicious logins that are inconsistent with the login structure of an enterprise network.

Specifically, we adopt a semi-supervised anomaly-based approach and build a one-class classifier for detecting malicious logins. To specify the class of normal logins, we propose a pattern mining algorithm to extract *login patterns*, each of which describes a subset of normal connectivities of network logins. Our algorithm uses a variation of the market-basket analysis algorithm [14]. Elements of a pattern include attributes of the connecting user (u), the source computer (s), and the destination computer (d) of login. Login patterns together define the *network login structure*. If a new login is not consistent with the login structure, we classify it as a malicious login.

In summary, this paper makes the following contributions:

- We explore the idea of detecting Credential-based Lateral Movements using login anomaly detection within an enterprise network.
- We propose a method for modeling login structure of enterprise networks using login patterns. We also provide an algorithm to automatically and efficiently extract login patterns from a large login data set.
- We evaluate our method using a real data set of logins and based on labeling by security analysts. Natural dynamics of network logins and changes in the organizations are sources of false positive in our system. We analyze these dynamic and provide insights for follow-up works aiming to improve the precision of the system. To the best of our knowledge, this work provides the first analysis of the structure and dynamics of logins within an enterprise network.

The rest of this paper is organized as follows. Section 2 overviews the system we have designed and its components. Section 3 describes our login pattern mining algorithm. Section 4 details the login classifier. In Section 5, we evaluate our method. This evaluation includes analysis of the dynamics of logins of a real network. It also includes measuring precision, true positive, and false positive of the classifier. Section 6 reviews related work. Finally, we discuss limitations of our current study and outline future work in Section 7. We conclude the paper in section 8.

2 OVERVIEW OF THE SYSTEM

Attackers use stolen credentials in different ways each of which might need a dedicated approach for detection. The focus of this paper is the detection of malicious logins that differ from the expected norm of a login concerning the user, source, or destination of it. In this section, we formalize the problem and detail the technique we have developed for detection.

2.1 Problem Statement

Credential-based Lateral Movement (CLM) is a network attack method in which an attacker uses a stolen credential to log in to a new computer to compromise it and therefore append it to a chain of hacked computers. An attack of this type usually starts with a phishing attack that compromises a user's workstation within an enterprise network [32]. The end goal of the attacker is to compromise computers that host high-value assets, such as a database

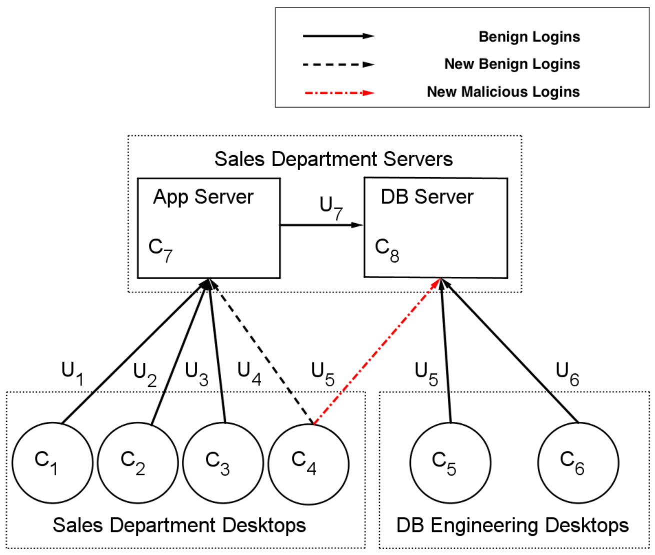


Figure 1: This diagram shows a simplified schematic of logins within a network. Solid lines represent logins that are observed in a time interval in the past. Dashed lines are new logins, one of which is benign (dashed black) and another of which is malicious (dotted red). The benign login is consistent with normal login structure, but the malicious login is inconsistent.

or an application server enabling critical operations. In his journey, from a workstation to a target server, the attacker continually steals new credentials and uses them to compromise a computer and extend the chain of compromised computers. We can describe a *state of an attack* using a set CC of compromised computers and a set CU of compromised user accounts (i.e., stolen credentials). In this context, a compromised computer is one that is owned by and located within an enterprise network, but being exploited by an attacker to run an arbitrary program (e.g., malware). Relying on compromised computers as stepping-stone, the next move of an attacker is to use a stolen credential u (i.e., $u \in CU$) to login from a compromised *source computer* s (i.e., $s \in CC$) to compromise a *destination computer* d that is not already compromised (i.e., $d \notin CC$).

As the attacker uses credentials to log in to computers, some of his login connectivities might be inconsistent with normal network logins concerning user account and computers involved in those logins. Such inconsistencies are inevitable because the attacker can only use computers and user/system accounts that he has already compromised, for his logins to computers that he wants to compromise. We leverage this observation to detect malicious logins.

2.2 System Architecture

Our approach for detecting malicious logins is anomaly-based and focuses only on identifying abnormal connectivities. The architecture of the system we developed is presented in Figure 2. Our

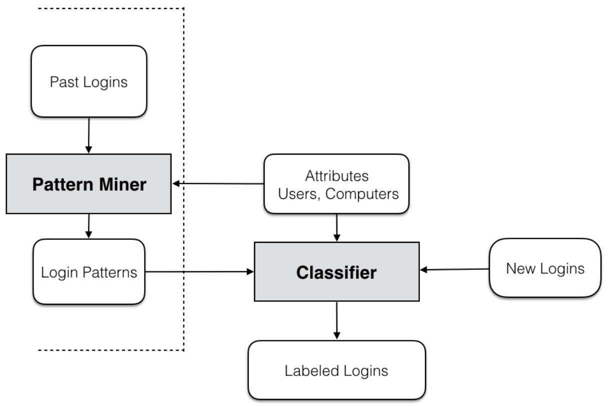


Figure 2: The inputs, processes, and outputs of our login anomaly detection system. The Pattern Miner component mines the login patterns. The Classifier component utilizes the login patterns to classify new logins into two classes of benign and malicious logins.

login anomaly detection system is composed of a *pattern miner* component and a *login classifier* component.

Pattern Miner. Our method of detecting malicious logins relies on their inconsistency with normal logins. Therefore, this method needs to model the normal logins within a network. Put another way; our algorithm needs to specify how users usually login between computers. To model these logins, we introduce the notion of a *login pattern* which describes a subset of network logins with regards to their connectivities. For example, it is intuitive to observe, based on the logins of Figure 1, that logins of Sales from a desktop in that department to the application server is a normal login pattern. We define a login pattern $\mathcal{P}$ as a set of attributes for both users and computers, and show it as a triplet of attributes $\mathcal{P} =< \widehat{U}, \widehat{S}, \widehat{D} >$. Examples of such attributes are the type (e.g., primary, admin or service account) and title (e.g., investment banking manager, help desk) for a user, and role (e.g., workstation or server), location, and type of application that a server computer hosts, for computers.

The role of the pattern miner component is to mine logins and extract login patterns. Inputs of this component are the history of all logins of an interval, spanning a few months in the past, and the attributes of both users and computers during the given time interval. After processing these logins and mining patterns, this component outputs a collection of login patterns as well as confidence scores that indicate their reliability. Structure of logins within an enterprise network, such as the financial company where we collected our data set from, are subject to change. Therefore, the pattern miner component should be scheduled to mine and update login patterns, periodically. The optimum frequency of updates depends on the pace of changes in the network login structure.

Login Classifier. Another component of this system is a classifier. New logins are one of the inputs to this classifier. It also uses normal login patterns extracted by the pattern miner component as input. By computing the similarity of the new login with the class of normal logins, this component classifies new logins into one of two classes; *benign* or *malicious*.

2.3 Adversarial Model

A typical attack that our system can detect has some characteristics. The main feature of such attack is that the attacker uses credentials to log in from one computer to another within a network. Our detection method is independent of how a credential is stolen. An attacker might steal a password using keyloggers as it is entered into a login page. He might also use tools such as Mimikatz [8] that directly access password areas in memory of computer and steal them as new user logs in locally or remotely. Regardless of the method of stealing a credential, our detection method detects attacks when the credential is used to login to computers over the network. These logins might occur by the attacker entering credentials into a login window or as a result of authentication based on cached credentials when using software (e.g., file sharing protocol) to access resources on a destination computer. In either case, network records an incident of login, and our system uses it for detection.

The credential that an attacker uses to log in from a source computer to a destination computer is crucial to our detection algorithm. If the attacker uses a credential that is normally used between two computers, our algorithm can not detect the malicious login. However, the attacker is not always able to satisfy this condition. For example, consider an attacker who steals the credential of a help-desk admin who logs in remotely to a compromised computer to fix it[1]. If the attacker uses this stolen credential to log in from the compromised computer to any other computer, he will be detected by our algorithm. It is also the case if the attacker tries to log in back to help-desk admin's computer since our algorithm considers the direction of logins as well.

An attacker who is aware of an implementation of our algorithm and knows normal login connectivities within a network environment does not necessarily have an advantage for evading detection. To exploit his knowledge and bypass our detection algorithm, the attacker must as well have the right credential and be located on a computer that usually logs in to a destination.

3 PATTERN MINING

The pattern miner is the core component of our system. It extracts login patterns each of which specifies a network login substructure. A pattern is composed of attributes of a user, a source computer, and a destination computer. Before describing the algorithm that mines patterns, we first define some terms.

3.1 Definitions

Login. A login of a user u from computer s to d is uniquely identified and presented by the triplet $l =< u, s, d >$. For example, in Figure 1, the network login of the user u_1 from the source computer c_1 to the destination computer c_7 can be represented by a triplet $< u_1, c_1, c_7 >$.

Login History. A collection of logins from a given period in the past composes a login history H. The pattern mining algorithm uses H to mine the login patterns.

Login Attributes. Each of the three elements of a login has some attributes. Therefore, a login can be represented by a triplet of

[1]Attackers sometimes slow down compromised computers to trick admins to log in to a compromised computer using their administrative credentials, to steal them.

Symbol	Description
$l =< u, s, d >$	A login, composed of user, source, and destination.
$L =< U, S, D >$	A login attribute, composed of attributes of each component.
$\mathcal{P} =< \widehat{U}, \widehat{S}, \widehat{D} >$	A login pattern, composed of some attributes of each component.
$< U^*, S^*, D^* >$	Power set of attributes.

Table 1: This table shows notations and description of symbols.

attributes in form of $A =< U, S, D >$ where $U = \{x | x \in A_u\}$ is the collection of all attributes of the user u, $S = \{y | y \in A_c\}$ is a collection of all attributes of the source s, and $D = \{z | z \in A_c\}$ is a collection of all attributes of the login destination. Each attribute describes one aspect of the login, including the role of user or computer, location, or type of computer. In the example of Figure 1, the login attributes of login $l =< u_1, c_1, c_7 >$ are $A =< (primary, Salesstaff), (Desktop, SalesDept), (Server, SalesApp) >$.

Login Pattern. A login pattern $\mathcal{P} =< \widehat{U}, \widehat{S}, \widehat{D} >$ describes a sub-structure of network logins. Each element of the pattern is composed of a subset of the attributes of users and computers. In the example of Figure 1, $\mathcal{P}_1 =< (Sales\ Staff), (Desktop, Sales\ Dept), (Sales\ App) >$ is a pattern.

Pattern Occurrence. We say that a login $l =< u, s, d >$ with attributes $A =< U, S, D >$ is an occurrence of the pattern $\mathcal{P} =< \widehat{U}, \widehat{S}, \widehat{D} >$ iff $\widehat{U} \subset U$, $\widehat{S} \subset S$, and $\widehat{D} \subset D$. For example, the login $< u_4, c_4, c_7 >$ is an occurrence of the pattern P_1. In comparison, the login $< u_7, c_5, c_8 >$ is not an occurrence of it. It should be noted that a login can be an occurrence of several login patterns.

Pattern Orientation. Depending on the ratio of number of users and computers of a type describing a pattern to all users and computers of that type, a login pattern can be categorized to *source-oriented*, *destination-oriented*, or *user-oriented*. Figure 3 shows several possible orientations for a pattern. Below, we describe each of these orientations:

- **Source-Oriented.** A pattern $\mathcal{P} =< \widehat{U}, \widehat{S}, \widehat{D} >$ is source-oriented if a noticeable fraction of source computers with attributes S have at least one pattern occurrence in login history H. An example of this orientation is the pattern describing logins of all employees of a department to a server hosting an application related to responsibilities of that department.
- **Destination-Oriented.** A destination-oriented pattern has a noticeable fraction of destination computers with attributes D with at least one pattern occurrence in H. An example of this orientation is pattern of logins of a patch management server that accesses several computers of a given type to push patches of an operating system or application.
- **User-Oriented.** A user-oriented pattern has a noticeable fraction of users with attributes U with at least one pattern occurrence in H. An example of this orientation is pattern of delegated logins of many users through proxy applications such as mobile gateways or exchange servers.

Orientation Score. Our algorithm computes a score for each of the three orientations. An orientation score represents the degree to which a pattern has an orientation. A login might have high scores for more than one orientation. For example, a pattern related to the logins of desktops to domain controllers has a high score for all orientations because all users and computers connect to the domain controllers since they are configured to work in a load-balancing manner. Later in this section, we will describe how our algorithm computes the orientation scores.

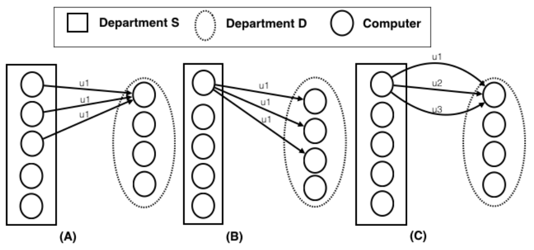


Figure 3: These figures demonstrate logins with the same patterns but varying orientations.

Login Graph	S-score	D-score	U-score
(A)	0.6	0.25	0.33
(B)	0.2	0.75	0.33
(C)	0.2	0.25	1

Table 2: Orientation scores of a pattern $\mathcal{P} =< \widehat{U}, \widehat{S}, \widehat{D} >$ with different orientations as shown in Figure 3. All users assumed to have the same attributes.

3.2 Pattern Mining Algorithm

Our pattern mining algorithm is similar to association rule mining in market-basket analysis algorithms [14]. It employs two steps to mine patterns of network logins. In the first step, it enumerates candidate login patterns from each login in the login history H. In the second step, this algorithm groups login patterns and counts the number of occurrences of each. It also computes their orientation scores. Finally, the algorithm selects patterns with orientation scores above a specified threshold. These selected patterns specify characteristics of the network's login structure and will be used for detecting anomalous logins.

Enumerating Candidate Patterns. To enumerate candidate login patterns, we first generate three power sets (i.e., the set of all subsets), each based on attributes of elements of login. We denote these power sets by U^*, S^*, and D^*. Then, we create the Cartesian product $U^* \times S^* \times D^*$ that generates all candidate patterns related to one login. We exclude candidate patterns that are missing all the attributes of any login element. Therefore, the number of possible login patterns generated based on a login is equal to $(|U^*| - 1) \times (|S^*| - 1) \times (|D^*| - 1)$. For example, for a login of a user ("Sales","Staff") from ("Desktop","Sales") computer to ("Sales-Dept","Server") (see Figure 1), the number of candidate patterns is 27 (three non-empty subsets of attributes for each element). The total count of unique candidate patterns based on all logins in H depends on the number of unique values of login attributes as well. Algorithm 1 shows a simplified implementation of this algorithm.

Algorithm 1 This algorithm generates all pattern candidates from a given login. The operator * computes power set of a given set.

```
1: procedure EnumeratePatterns($u$, $s$, $d$)
2:     get $< U, S, D >$
3:     gen-powerset $< U^*, S^*, D^* >$
4:     for $\widehat{U} \in U^*$ do
5:         for $\widehat{S} \in S^*$ do
6:             for $\widehat{D} \in D^*$ do
7:                 emit-candidate ($< \widehat{U}, \widehat{S}, \widehat{D} >$)
```

Computing Orientation Scores. To identify the orientations of a pattern $\mathcal{P} =< \widehat{U}, \widehat{S}, \widehat{D} >$, we calculate three orientation scores for each pattern, as follows:

- **S-score.** This score represents the source orientation of a pattern $\mathcal{P}$. We compute the ratio of computers that satisfy attribute S and appear in an occurrence of the pattern $\mathcal{P}$ in the login history H to the count of all computers that satisfy attribute S.
- **D-score.** This score represents the destination orientation of a pattern $\mathcal{P}$. We compute the ratio of computers that satisfy attribute D and appear in an occurrence of pattern $\mathcal{P}$ in the login history H to the total number of computers that satisfy attributes D.
- **U-score.** This score represents the degree to which a pattern $\mathcal{P}$ is user oriented. We compute the ratio of users that satisfy attribute U and appear in an occurrence of the pattern $\mathcal{P}$ in the login history H to the total number of users that satisfy attribute U.

Table 2 shows three orientation scores of patterns presented in Figure 3.

3.3 A Fast Algorithm for Pattern Mining

A major task of our algorithm is to extract the candidate login patterns and compute the Cartesian product of the power sets for the attributes of each login. The time complexity of these computations over sets of values are non-polynomial, and therefore are very expensive. Also, the total number of unique login patterns extracted from a real dataset of login attributes can be overwhelming. For example, our algorithm generated 2.3 billion candidate patterns from a dataset of more than 600,000 unique logins where nine attributes described each login. The reason for this number of candidate patterns is that each login attribute has several possible values and therefore there are several possible combinations. For example, in the dataset we studied, the location of computers has 70 possible values, each of which indicates a site of the global financial company where a computer located. Considering this volume of patterns to process, Algorithm 1 is not scalable. In this section, we describe our technique to tackle this challenge.

Overview. To create a fast and scalable algorithm for generating the candidate login pattern of a big dataset, we follow the encoding and parallelization process illustrated in Figure 4. This process includes encoding to minimize the memory required to represent the patterns, and parallelization to improve the speed of execution by a divide and conquer approach.

Encoding. Our approach to reduce the memory required to store the Cartesian product of power set of attributes is a binary encoding of the attributes of users and computers. The proposed encoding assigns an integer code to each value and generates a binary mask for each different combination of these attributes. Using this method, we present a login entry using the attribute codes. This encoding takes considerably less space than storing string values. More importantly, the login patterns only include attributes that describe a pattern, and the binary mask identifies which code belongs to which attribute. This compresses the space required to store each pattern.

Parallelization. After reading logins and encoding their attributes, the algorithm for generating patterns creates required masks. The number of these masks is equal to $2^{|U|+|S|+|D|}$. For example, if total number of attributes of login elements is nine, then 512 mask values, ranging from 0 to 511, will be generated. Our parallelization method splits these masks into several clusters, each assigned to a CPU core for processing. Collectively, these parallel processes generate all pattern candidates and output them into file storage. We used Spark [1] for parallelization and Python generators to improve the speed of the pattern generation algorithm.

Figure 4 shows flow of the pattern generation algorithm. For a login $L_i =< U_i = (User1, DPT1, GB), S_i = (C1, BLD1, LN), D_i = (C2, BLD2, NY) >$, our algorithm first encodes the string values, say $User_1$ to 1 ($User_1 \longrightarrow 1$) and DTP_1 to 3 ($DTP_1 \longrightarrow 3$), etc. After that, the pattern generator creates the power set of the encoded login attributes. For example, for storing $< (\{\}, \{\}, 2), (\{\}, \{\}, 2), (\{\}, \{\}, 1)) >$ pattern, binary mask 73 (binary 001001001) will be used. Using this encoding, the compressed format of the pattern which is $73 : 2, 2, 1$ will be stored. This compacted presentation reduces the space required to store generated patterns, dramatically.

For parallelization, our system runs the pattern processing algorithm in a separate cluster for each range of mask values. This parallelization accelerates the pattern mining algorithm to extract patterns within minutes for a big dataset of logins.

4 LOGIN CLASSIFIER

The classifier of our system is a hybrid of two components and evaluates each login independently. The first component uses a *exact matching* approach and the second one uses *pattern matching* for classification.

Exact Matching. The exact matching classifies a login $l =< u, s, d >$ as benign if there is a login $l' =< u', s', d' >$ in the login history H, where $u = u'$, $s = s'$, and $d = d'$. Otherwise, it classifies the login as malicious. An attacker may bypass this classifier by poisoning the login history used for classification. To reduce this possibility, we

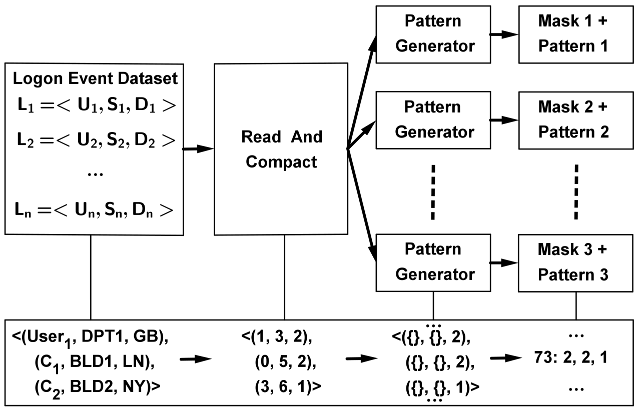


Figure 4: This diagram shows the process of encoding and parallelization of the algorithm for generating the candidate login patterns.

exclude infrequent logins from the login history. To be included the login history, a login must occur a sufficient number of times (e.g., minimum 10% of days) during the time interval that the system collects logins. Therefore, an attacker will not be able to contaminate the login history H without many logins that increase the risk of detection.

Pattern Matching. A pattern matching classifier first generates all possible combination of attributes related to a login L with attributes $A =< U, S, D >$ using the same approach used for enumerating candidate network login patterns. The classifier classifies the login as benign if at least one of the combinations of login attributes matches a pattern of the set of network login patterns that describe the network structure. In other words, login $l =< u, s, d >$ will be classified as benign if it is an occurrence of one of the patterns describing the network login structure. For example, the login $< u_4, c_4, c_7 >$ in Figure 1 is an occurrence of the pattern $\mathcal{P}_1$ and therefore will be classified as a benign login. In contrast, the login $< u_7, c_5, c_8 >$ is not an occurrence of any of the patterns and consequently will be classified as malicious. The advantage of pattern matching over exact matching is that it is flexible concerning legitimate changes of network logins. In fact, many new logins do not exactly match a previous benign login but match normal login patterns.

In addition to pattern matching, our algorithm computes a confidence score for each benign login that does not exactly match any past benign login but matches a normal login pattern. The confidence score is computed with respect to the difference(s) with all other occurrences of that pattern. For example, a new login might connect to an instance of a type of destination computer type $\widehat{D}$ that none of the other logins that match a pattern connect to it. In this case, our algorithm uses the destination orientation of the pattern as the confidence of matching a login with normal logins. Other orientation scores will be used accordingly. Our algorithm uses the minimum orientation score of a pattern if all three elements of a login are different from past logins that match a normal login pattern.

5 EVALUATION

We evaluated our system using a real data set of all logins of a global financial company for a duration of five months. In this section, we first explore the network structure and dynamics of logins of the enterprise network. Then, we measure the *precision* of alerts generated by our algorithm based on manual labeling by a number of security analysts. Finally, we evaluate the *false positives* and *true positives* of our algorithm based on its performance over synthetic attack traces injected into the data set of real logins.

5.1 Dataset

The dataset of logins we used includes all login entries each of which identifies a unique login event between two different computers. A login connection includes username, and name of source, and destination of a login. For each login, the dataset also includes the daily count and type of login. A login type indicates the protocol used for authentication. One of the login types is *Windows network login*. This is the most common login type and happens implicitly (using cached credentials) or explicitly (asking for username and password), with or without user interaction, and in different scenarios, including using file sharing&printing services and connecting to a Kerberized application[2]. Another login type is *remote interactive login* (i.e., Remote Desktop) that is used when a graphical interface connects to another computer.

The login data set includes some attributes for each of the three elements describing a login. We collected these data from different databases of the company, including HR and Change Management databases. For each user, two attributes of type-of-user (e.g., end user/admin/service account) and business unit (e.g., investment banking, information technology) were available. For each computer, three attributes of type-of-computer (e.g., desktop/server), application (e.g., domain controller/exchange server/HR software), and location of the computer (e.g., New York[3]) were provided. Table 3 lists the attributes of each login element.

The logins in the data set involve a total of 25,450 unique usernames. A total of 12,550 usernames belong to computer accounts. These accounts are created to connect a computer to a Windows network. The remaining usernames belong to individuals, admins and non-admin users, and services.

The data set involves 33,151 unique computer names. These computers mostly are desktops and servers with a permanent name. A subset of the computer names, however, belongs to virtual desktops that acquire a name on the spot when users initialize them. The average number of unique login connections (i.e., unique triplets of the user, source, and destination) was about 160,000 logins per day.

Login Connectivities. The number of unique login connectivities in this data set is 633,657. That is 0.05% of all possible connectivities between different computers. This small percentage confirms that legitimate connections between computers are structured. In fact, client/server interactions are the major use case in enterprise networks. As a result, logins are mostly to servers, and desktops infrequently login to each other. To confirm, we obtained the average in-degree (i.e., login to) and out-degree (i.e., login from) of

[2]An application accepting Kerberos tokens for authentication.
[3]Truncated for the privacy of the company.

Component	Attributes
User	Type, Business Unit
Source Computer	Type, Application Name, Location
Destination Computer	Type, Application Name, Location

Table 3: Attributes of login elements used for pattern mining.

computers in the dataset. While average in-degree of servers is 293, it is 14 for workstations.

To understand the login connectivities further, we studied organizational structure and its correlation with the login connectivities. For example, we examined the logins of two business units A and B[4] with 82 and 17 employees, respectively. We observed that none of the users of the department A logged in from a computer of the department B, and vice versa, during five months. We also observed that users of business unit B connected to 18 computers (local servers) to which group A did not connect and that computers of both groups logged in to 13 global servers (e.g., domain controllers and email servers) that provide service for the entire network. These imply that enterprise logins are structured. In other words, we can predict which user logs in from a computer to another one. The goal of our system is to capture and model the login structure of a network and use that for anomalous loging detection.

Dynamics of Logins. We analyzed the data set to understand the dynamics of logins. More specifically, we studied the changes of login connectivities in comparison to the history of a collection of logins in the past. Compared with logins of history, we categorize each login into five different classes:

- **No-Change.** A login represented as a triplet $l =< u, s, d >$ (i.e., user, source and destination names) is a no-change login if there exists a login $l' =< u', s', d' >$ in the history of logins where $(u = u')$, $(s = s')$, and $(d = d')$. In other words, if the login l has happened in the past, then it is a no-change login.
- **Source-Change.** If a login $l =< u, s, d >$ has not happened in the past but there exists a login $l' =< u', s', d' >$ in the history of logins where $u = u'$, $s \neq s'$ and $d = d'$, then the login is categorized as source-change. In Figure 5, the login $< u_3, s_2, d_2 >$ is a source-change compared with history of logins. This type of change happens for a couple of reasons, including logins of employees from a different office while traveling, or using virtual desktop technologies when working from home.
- **Destination-Change.** If a login $l =< u, s, d >$ has not happened in the past but a login $l' =< u', s', d' >$ exists in the history of logins where $u = u'$, $s = s'$ and $d \neq d'$, then the login is categorized as destination-change. In Figure 5, the login $< u_2, s_2, d_2 >$ is destination-change. This category of logins happens for several reasons, including connection to a new server in a collection of load balanced servers, deployment of new servers, activation of a disaster recovery process, or assignment of a task to an employee, that requires access to a new server computer.
- **User-Change.** If a login $l =< u, s, d >$ has not happened in the past but there exists a login $l' =< u', s', d' >$ in the history of logins where $u \neq u'$, $s = s'$ and $d = d'$, then the login is categorized as user-change. In Figure 5, the login$< u_3, s_2, d_1 >$ is an user-change, because computers u_2 and d_2 have been previously connected using user u_2, but not u_3. Such a change happens for several reasons, including delegating logins between proxies and destination servers.
- **Full-Change.** If a login $l =< u, s, d >$ has not happened in the past and none of the pairs of login elements have occurred in the past, then this login is a full change compared to the history of logins. This type of change usually happens because of the addition of a new user or computers to the network, or because of a significant shift in the roles of users or computers. In Figure 5, the login $< u_1, s_2, d_2 >$ is a full-change.

[4]Redacted to protect sensitive information.

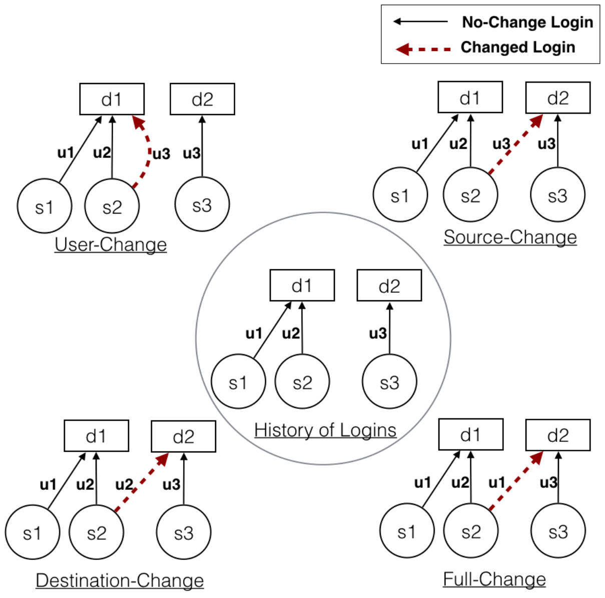


Figure 5: This diagram shows four categories of login changes. Logins at the center represent history of logins in the past.

We measured the change of logins concerning the categories described above. For this purpose, we used four months of logins to compile an aggregated history of logins. Then, we identified the type of change for every login for the month after. Figure 6 shows the percentage of logins of each category. The category of no-change is dominant and covers 85-95% of logins. The rate of logins that do not change shows that most of the logins of an enterprise network have happened at some point in time in the past. Another observation is that the percentage of no-change decreases as time elapses. This drift is due to long lasting changes in the network (e.g., a new server) and business operations. Our analysis shows that non-admin users are responsible for about 70% of the changes, admins for 20%, and service accounts for 10% of the changes.

The most common category of changes was destination-change which covers 2-6% of all logins. About 80% of the destination-changes involved a server as the login destination. Our analysis shows that between 1-2.5% of all logins include a source-change. Servers and desktop account for equal portions of source-changes. User-change and full-change were the smallest categories of login change.

Based on these statistics, we conclude that the majority of current logins in a network have been observed sometime in the past. Moreover, there are logins that we have not seen in the past but legitimately occur because of natural dynamics of network and organization. The goal of our technique is to distinguish benign from malicious login changes.

5.2 Experiment by Security Analysts

In the absence of a set of labeled bad logins, we asked some security analysts of the same financial organization that provided the login data for us to evaluate the logins that our system detects as malicious. In this section, we describe the setup of the system, methodology of evaluation, and the results of this experiment.

System Setup. The classifier of our algorithm has two components each requires data for training. The first component is an exact matcher that uses a history of logins from the past. We prepared the input for this component by aggregating four months of previous logins.

To enable the second component, the pattern matcher, the system first needed to mine the login patterns. Therefore, we input the logins of the past four months as well as details of computers and users to the pattern mining algorithm. Overall, eight different attributes, including two for users and three for each computer was used to describe login attributes. The pattern mining algorithm generated about 200,000 patterns. Also, it computed three orientation score for each login pattern. We provided the entire logins of one day to the system to classify the logins. The total number of logins given was about 177,000 and system generated 578 alerts.

Labeling Process. Since labeling the entire set of alerts was resource intensive, we asked the security analysts to label only a subset of alerts. We scheduled half-hour sessions with three analysts and asked each to label the logins while thinking aloud describing the reason for assigning each label. We developed and used an online labeling panel that shows only one login at a time to minimize the bias of showing more than one alert. Each security analyst used his own workstation to visit the online labeling page. They also were allowed to use other security and information management software they usually use, to cross-verify their hypothesis in case they weren't sure about their answer or needed to find an explanation. For each alert, we asked them to mark one of the following:

- **(a)** The login is most likely a good login
- **(b)** This needs further investigation
- **(c)** The login is most likely a malicious login

We considered the labeling of the security analysts as the ground truth. If a security analyst chose option (a), we would consider the login as benign. Making sure that a login is malicious requires efforts by security analysts, that were beyond resources allocated to this experiment. Moreover, labeling a login as (b) indicates that the login is unusual and potentially suspicious. Therefore, we consider logins marked either (b) or (c) a suspicious login.

For each alert, we showed the username, name of the source and destination computers, and all eight attributes related to user and computers. After labeling an item, the system showed another login for labeling. During the labeling, the analysts were allowed to consult other sources of information, including an HR system describing more details of responsibilities for a user, along with a database that provides detailed information about computers in the network.

Precision of the System. Precision is a measure of the positive predictive value of a classifier. For an attack detection system, it is computed as the fraction of true alerts among all alerts generated by a system. In other words, the precision of a binary classifier is computed by $\frac{TP}{TP+FP}$ where TP and FP are the counts of true positive and false positive, respectively. This measure is specifically suitable when the actual number of positive samples in the dataset is uknown.

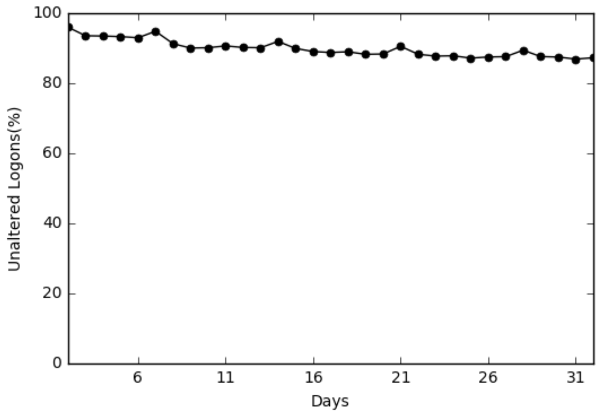
To measure the precision of our system, we showed an overall of 80 alerts randomly selected from 578 alerts generated by the system to three security analysts. We showed an almost equal number (i.e., one-third) of alerts to each one. Each alert was labeled only by one security analyst. We asked them to go through each alert one by one and label them. Security analysts marked a total of 11 out of 80 of logins as suspicious. Therefore, the accuracy of our system was 13% in the described experiment. Low precision is a canonical problem of anomaly-based detection approaches [2, 31].

We investigated the reasons for low accuracy of our system by seeking feedback from security analysts. The main reasons for false alerts were (I) logins related to admin accounts, and (II) missing attributes of elements of login. The reason that our system generates false alerts for many admin logins is the difficulty of modeling behavior of admins. In most of the cases, legitimate logins of admins are not frequent enough and therefore do not pass the threshold to be include in the history of login. As a result, a pattern will not be generated to cover them. Details of attributes for new users and computers are not quickly updated in the database. That is the main reason for missing attributes of elements of some login (i.e., user and computers). Without them, our system is not able to do the pattern matching and therefore identifies them as suspicious logins.

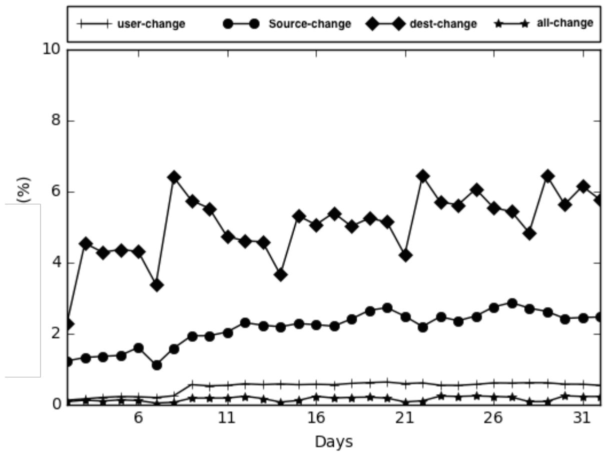
5.3 Experimenting with Synthetic Attack traces

Our evaluation based on feedback from security analysts was limited mainly because without knowing the actual number of malicious logins we can not measure the number of malicious logins that our algorithm misses. To overcome this problem, we evaluated our system based on several synthesized malicious logins injected into real traces of logins from the enterprise network.

Benign logins. We used five consecutive months of login data set to set up and evaluate our approach. We split this dataset into two



(a) No-Change Trend



(b) Login Changes (*User-Change, Source-Change, Destination-Change, Full-Change*)

Figure 6: (a) shows the percentage of no-change logins relative to a collection of logins of the past four months. (b) shows the percentage of each category of login change. Most of the changes are related to destination-change. The y-axis is shortened for readability.

partitions. We used four months of logins to train our anomaly detection algorithm to extract the login structure of the networks. With the last month of logins, we measured the false positive rate of our algorithm.

Attack traces. We generated traces of malicious logins based on a realistic strategy of attack that is often employed by penetration-testing campaigns to emulate a lateral movement attack within an enterprise network. Accordingly, we consider the following assumptions about an attack of this type:

- The attacker has already infected and compromised a workstation within an enterprise network and tries to transmit to another computer.
- The attacker is able to steal the password of any user who is active on a compromised computer. This includes accounts that are used to login from or to the compromised computer.
- The transmission of an attacker is naturally constrained by standard access controls of a network. The most important one is that non-admin accounts can only gain administrative access to a workstation. To compromise a server, the attacker must have access to an admin account.
- A stealthy attacker minimizes the number of malicious logins, as too many logins can raise an alert. Therefore, attackers that we simulated tried to login only to five other computers in the network.

To generate some malicious login traces, we randomly selected a number of workstations as a source of malicious logins. Then, we identified all credentials that were active on each computer and marked them as compromised. These were credentials that an attacker could use to log in and compromise other computers. Then, we chose five random target computers as the destination of a malicious login from each compromised computer. Using this process, we generated 150 traces of malicious logins in the form of $< u, s, d >$. This set of logins comprised the positive samples of our data set. We injected these malicious logins into the data set of logins of the enterprise and used it for measuring the performance of the algorithm.

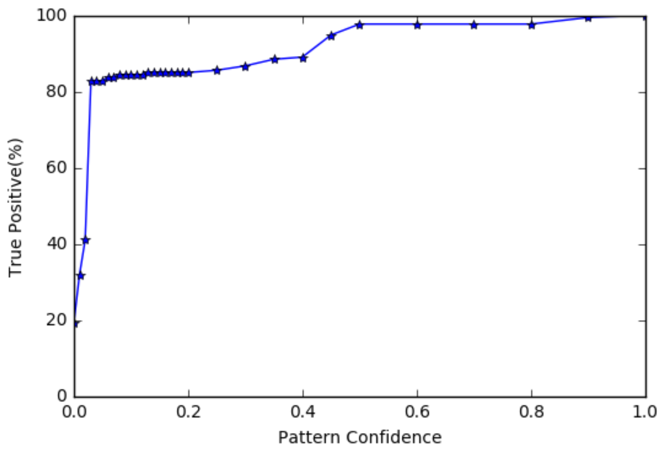


Figure 7: True positive rate of the algorithm based on different threshold for pattern confidence.

Quality of Patterns. Performance of our detection algorithm depends on the quality of patterns used for classification. One measure of the quality of patterns is their confidence scores. Patterns with higher orientations scores are more reliable and therefore have higher confidence. Here, we report the performance of the system concerning different thresholds of pattern confidence.

True Positive Rate. An important performance measure of a detection algorithm is true positive rate that is the capability of the

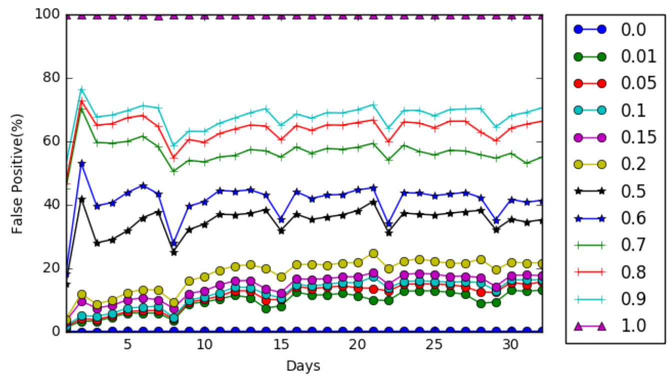


Figure 8: False positive rate relative to different thresholds for pattern confidence.

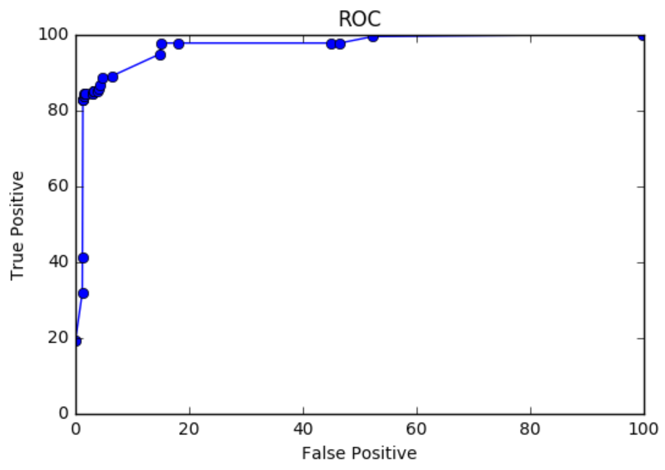


Figure 9: ROC diagram shows the false positive vs. True positive of the detection method.

system to detect malicious logins. The true positive rate is computed using $\frac{TP}{TP+FN}$. To measure this, we counted the number of malicious logins of the test data set that the classifier was able to detect. The true positive rate of the algorithm was varied depending on the minimum confidence score to include a login pattern in the classifier. A higher threshold for orientation scores results in patterns with greater confidence. It is less likely that our algorithm matches a malicious login patterns with a legitimate login pattern mistakenly when it uses a higher confidence threshold. Figure 7 shows the percentage of true positives relative to the threshold of pattern confidence. As our system increases the threshold, it will be able to detect more malicious logins.

False Positives Rate. A detection algorithm is only usable if the number of false alarms it generates is manageable. Number of false alerts that this system generates varies depending on the pattern confidence threshold. If the system uses a lower threshold, it will include more login patterns. Therefore, it will be more likely to find a matching pattern with a new benign login and this results a lower false positive rate. Figure 8 shows the false positive rate of the algorithm at different thresholds. We observe that false positive rate decreases as the system use lower threshold for pattern confidence.

ROC Curve. The threshold of pattern confidence that the system uses for choosing qualified patterns affects its false positive and false negative rates; While increasing the threshold improves the true positive rate, it increases the false positive as well. To show this interaction and find a balancing threshold, we used receiver operating characteristic (ROC) curve. In a ROC curve, the X-axis shows the false positive rate of the classifier and the Y-axis represents its true positive rates. We computed the data points based on different thresholds of pattern confidence. Figure 9 shows the ROC curve of our classifier. This diagram indicates that our system is able to detect 82% of malicious logins, while generating 0.3% false alerts.

6 RELATED WORK

This section reviews previous works relevant to ours. This section also clarifies the scope of our work and reasons why some alternative approaches might not be suitable for detecting CLM.

Role-based Access Control. Role-Based Access Control (RBAC) defines rules for access to network resources based on the role of users. Several mechanisms such as Access Control Lists in Linux and Active Directory in Windows systems [19] allow the network admins to enforce rules of access. Using the notion of groups of users and objects, network admins can allow or deny access of a group of users to a collection of resources. This mechanism is useful for stopping an employee from accessing data or resources he/she should have *never* access. For resources that a user might need access, the access is granted even if the user needs it rarely. In fact, business continuity is the main reason for granting more access than is required at each particular point in time. As a result, 50-90% of users are over-entitled [4]. These excessive permissions allow an attacker to move almost freely within a network. This work complements the access control mechanisms and is suitable for an enterprise environment where business continuity has high priority. Our system generates an alert if a login is abnormal. By investigating the alerts, security analysts can make sure that a highly unlikely login is not malicious.

Monitoring Activities within Networks. In response to strengthened networks and servers that resist direct external attacks, attackers have shifted to indirect attack methods. In one such method, the attackers compromise a desktop within a network using a phishing attack [5, 22, 34]. Then they use this foothold to compromise other computers or servers that host valuable data they could not access otherwise. This attack method motivates the production of monitoring and detection tools based on malicious traffic within enterprise networks [11, 23, 23, 35]. These tools rely on an enormous amount of data collected from network and host activities using sensors installed on computers and networking devices. Some detection approaches have only focused on infected computers. Yen et al. [35] have proposed a system that automatically mines knowledge from the log data produced by a broad range of security products (e.g., anti-virus, firewall) to detect infected workstations. Fawaz et al. [11] have proposed a framework for fusing data from different sources within a network to detect orchestrated attacks, including lateral movement. Oprea et al. [23] proposed a belief propagation technique that determines the state of a computer (i.e., benign vs. malicious) given prior knowledge about its past state and interactions with external resources (e.g., external websites). Using this technique, they have been able to discover new malicious entities. These techniques, do not utilize information about credential usage, and therefore can not detect the important class of credential-based lateral movement that we explored in this paper.

Detecting Malicious User Activities. Even though attackers use remote exploits and zero-day vulnerabilities, these methods are overrated [24]. Instead, attacks based on credential-based lateral movement (CLM), using usernames and passwords to move laterally between computers within a network [3], has prevailed. Some previous works have studied credential-based attacks. GonÃğalves et al. [13] employed credential usages for detecting misbehaving computers based on an unsupervised clustering approach. They used features such as the number of successful and failed logins, as well as statistics about admin logins for detection. Their approach is not able to detect CLM because it does not exhibit any statistical abnormalities such as frequent logins. In comparison, our approach can identify a single login of an attacker because it relies on the structure of logins instead of the frequency of them. Freeman et al. [12] have proposed a supervised statistical method for classification of logins in the client-server interactions. They use several features, including IP reputations to classify benign and malicious logins. In comparison, our method is related to logins within an enterprise network. These logins involve a more complex set of interactions between machines beyond client-server structure. Our approach is also different from theirs as we do not need labeled data for training our classifier. Instead, we use a semi-supervised anomaly detection approach. Siadati et al. [28] have used a signature-based method for detecting malicious logins. Their system relies on iterative visual exploration of logins within enterprise to identify and define signature of malicious logins.

Our approach has potential application in fraud and insider threat detection. Eberle et al. [9] have proposed a graph-based detection method for identifying anomalous actions concerning the interactions of computers within a network. Their approach computes the changes of a graph of interactions in comparison with a model of interaction they build atop the most frequent subgraphs of the connections. However, it is not able to correctly distinguish benign changes that occur due to network dynamics from malicious ones.

Evaluation Methods. Depending on the availability of appropriate test data, different methods can be used for evaluating anomaly detection approaches [10, 17, 21, 25]. The ideal scenario is when the ground truth is available [18, 31]. Similar to [15], we compiled a limited data set of labeled logins based on labeling of some security analysts and measured the precision of our system based on that. We also used another accepted method, that is creating synthetic attack traces and injecting the traces into the benign data [6, 7, 20, 27, 33].

7 LIMITATIONS AND DISCUSSIONS

Evasions. Our method utilizes network login structure to detect malicious logins that are not consistent with the normal login structure. An attacker who is aware of an implementation of our system and identifies login structure of a target company might try to evade detection. To imitate a legitimate login, the attacker must have the right combination of username and computer to log in to a destination computer. It is not always feasible for an attacker to satisfy these conditions. At the beginning of a lateral movement attack, it is more difficult for an attacker to satisfy these conditions and evade detection because the attacker has fewer compromised computers and stolen credentials. Therefore, our system will be able to detect attacks in their earliest stages. An attacker may combine CLM method with a vulnerability-based lateral movement to evade detection. Therefore, it is highly recommended to use our approach along with ones that detect remote software vulnerabilities.

Poisoning Attack. For training our classifier, we use logins in a period in the past. It is possible for an attacker to create some logins with the goal of misleading the pattern miner module to include an illegitimate pattern in the set of login patterns. To avoid this type of poisoning, we only use logins that have occurred frequently enough in the past. More specifically, we compute the percentage of the days in which a login has happened in the past. We include a login in the training only if this ratio is above a certain threshold. In our experiment, we used logins that happen in more than 10% of days in the past. Even if an attacker does a particular login more than 10% of the days, this event does not simply suggest a new pattern to our system. Instead, he must log in from enough number of different source computers of the type to enough number of destination computers of the type using appropriate usernames to suggest patterns with a minimum required orientation scores. Therefore, an attacker will not be able to contaminate the training data without risking detection.

Limitations. Although we had access to a rare data set of millions of logins of an enterprise, our study is still limited to one company and one type of enterprise. We are fully aware of the limitations of generalizing the findings of this paper to other networks, particularly those with more login dynamics and very different structures. Specifically, the stability of login structure varies from enterprise to enterprise. For example, in a development environment such as a software company, the login structure varies over time dramatically as some changes on a project might introduce significant changes in the login structure. Therefore, the updates of the classifier can not keep up with the dynamics of the network.

Our approach can not correctly classify some of the benign logins caused by dramatic changes in a network, such as activation of a disaster recovery center. A potential fix for this problem requires an intervention of security analysts to whitelist such login patterns prior or during the disaster recovery process. Another possible fix is to train a separate model at the time of the disaster recovery test and use the system as a disaster occurs.

We have not studied the effect of a longer period of logins as input for the pattern miner because we did not have access to such data. For the same reason, we have not studied the optimal window at which the algorithm should be retrained. However, according to the amount of change in the logins from 5% to 15% after a month, it is recommended to retrain the algorithm frequently. Our fast algorithm makes it easy to retrain the algorithm in a reasonable amount of time.

Our work, similar to any other work in the intrusion detection domain, suffers from canonical challenges in anomaly detection.

One of such problems is base-rate fallacy [2] that inherently challenges the precision of detection systems. Our strategy to reduce the amount of error was combining two classifiers; an exact matcher and a pattern matcher. To improve this even further, we plan to integrate the result of manual inspections of the security analysts in a feedback loop to the system and adopt an online learning process to reduce false positives and improve precision over time.

8 CONCLUSION

To the best of our knowledge, this is the first work reporting the internals and login dynamics of an enterprise network. We utilized the insights gained from our analysis to develop the notion of network login structure to model normal logins of enterprise networks based on triplets of attributes of user and computers involved in network logins. We have developed a fast and scalable pattern mining algorithm to automatically extract such login patterns. Utilizing the network login structure to model the class of benign logins, we built a binary classifier to detect structurally anomalous logins. We evaluated our system based on labeling of security analysts as well as synthetic attack traces. Our evaluation shows that using an appropriate data set for training, our system can detect more than 82% of malicious logins with 0.3% false alerts.

9 ACKNOWLEDGEMENT

Authors of this paper are thankful for valuable comments of anonymous reviewers of this article and proofreading and editing by Lois DeLong. We also thank Denis Khryashchev and Huy T. Vo for helping with implementation of an early version the fast pattern mining algorithm.

REFERENCES

[1] APACHE. 2017. Spark: A lightning-fast cluster computing. https://spark.apache.org. (2017).

[2] Stefan Axelsson. 2000. The base-rate fallacy and the difficulty of intrusion detection. *ACM Transactions on Information and System Security (TISSEC)* 3, 3 (2000), 186–205.

[3] Schneier B. 2016. Credential Stealing as an Attack Vector. https://www.schneier.com/blog/archives/2016/05/credential_stea.html. (2016). [Online; accessed 15-Feb-2017].

[4] Schneier B. 2016. Real-World Access Control. https://www.schneier.com/blog/archives/2009/09/real-world_acce.html. (2016). [Online; accessed 19-May-2017].

[5] Businessinsider. 2014. How The Hackers Broke Into Sony And Why It Could Happen To Any Company. http://www.businessinsider.com/how-the-hackers-broke-into-sony-2014-12. (2014).

[6] Baris Coskun, Sven Dietrich, and Nasir Memon. 2010. Friends of an enemy: identifying local members of peer-to-peer botnets using mutual contacts. In *Proceedings of the 26th Annual Computer Security Applications Conference*. ACM, 131–140.

[7] Kaustav Das and Jeff Schneider. 2007. Detecting anomalous records in categorical datasets. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 220–229.

[8] Benjamin DELPY. 2014. A little tool to play with Windows security. https://github.com/gentilkiwi/mimikatz. (2014).

[9] William Eberle, Jeffrey Graves, and Lawrence Holder. 2010. Insider threat detection using a graph-based approach. *Journal of Applied Security Research* 6, 1 (2010), 32–81.

[10] Hadi Fanaee-T and Joao Gama. 2014. Event labeling combining ensemble detectors and background knowledge. *Progress in Artificial Intelligence* 2, 2-3 (2014), 113–127.

[11] Ahmed Fawaz, Atul Bohara, Carmen Cheh, and William H Sanders. 2016. Lateral Movement Detection Using Distributed Data Fusion. In *Reliable Distributed Systems (SRDS), 2016 IEEE 35th Symposium on*. IEEE, 21–30.

[12] David Mandell Freeman, Sakshi Jain, Markus Dürmuth, Battista Biggio, and Giorgio Giacinto. 2016. Who Are You? A Statistical Approach to Measuring User Authenticity. In *NDSS, The Internet Society*.

[13] Daniel Gonçalves, João Bota, and Miguel Correia. 2015. Big Data Analytics for Detecting Host Misbehavior in Large Logs. In *Trustcom/BigDataSE/ISPA, 2015 IEEE*, Vol. 1. IEEE, 238–245.

[14] Jochen Hipp, Ulrich Güntzer, and Gholamreza Nakhaeizadeh. 2000. Algorithms for association rule miningâĂŤa general survey and comparison. *ACM sigkdd explorations newsletter* 2, 1 (2000), 58–64.

[15] Jaeyeon Jung, Vern Paxson, Arthur W Berger, and Hari Balakrishnan. 2004. Fast portscan detection using sequential hypothesis testing. In *Security and Privacy, 2004. Proceedings. 2004 IEEE Symposium on*. IEEE, 211–225.

[16] Krebsonsecurity. 2014. Target Hackers Broke in Via HVAC Company. http://krebsonsecurity.com/2014/02/target-hackers-broke-in-via-hvac-company/. (2014).

[17] Anukool Lakhina, Mark Crovella, and Christophe Diot. 2004. Diagnosing network-wide traffic anomalies. In *ACM SIGCOMM Computer Communication Review*, Vol. 34. ACM, 219–230.

[18] Richard Lippmann, Joshua W Haines, David J Fried, Jonathan Korba, and Kumar Das. 2000. The 1999 DARPA off-line intrusion detection evaluation. *Computer networks* 34, 4 (2000), 579–595.

[19] Alistair G Lowe-Norris and Robert Denn. 2000. *Windows 2000 active directory*. O'Reilly & Associates, Inc.

[20] Matthew V Mahoney and Philip K Chan. 2003. An analysis of the 1999 DARPA/Lincoln Laboratory evaluation data for network anomaly detection. In *International Workshop on Recent Advances in Intrusion Detection*. Springer, 220–237.

[21] George Nychis, Vyas Sekar, David G Andersen, Hyong Kim, and Hui Zhang. 2008. An empirical evaluation of entropy-based traffic anomaly detection. In *Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*. ACM, 151–156.

[22] NYTimes. 2014. Neglected Server Provided Entry for JP-Morgan Hackers. http://dealbook.nytimes.com/2014/12/22/entry-point-of-jpmorgan-data-breach-is-identified/. (2014).

[23] Alina Oprea, Zhou Li, Ting-Fang Yen, Sang H Chin, and Sumayah Alrwais. 2015. Detection of early-stage enterprise infection by mining large-scale log data. In *Dependable Systems and Networks (DSN), 2015 45th Annual IEEE/IFIP International Conference on*. IEEE, 45–56.

[24] Joyce R. 2016. USENIX Enigma 2016 - NSA TAO Chief on Disrupting Nation State Hackers. https://www.youtube.com/watch?v=bDJb8WOJYdA. (2016). [Online; accessed 15-Feb-2017].

[25] Benjamin IP Rubinstein, Blaine Nelson, Ling Huang, Anthony D Joseph, Shing-hon Lau, Satish Rao, Nina Taft, and JD Tygar. 2009. Antidote: understanding and defending against poisoning of anomaly detectors. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*. ACM, 1–14.

[26] Jerome H Saltzer. 1974. Protection and the control of information sharing in Multics. *Commun. ACM* 17, 7 (1974), 388–402.

[27] Taeshik Shon, Yongdae Kim, Cheolwon Lee, and Jongsub Moon. 2005. A machine learning framework for network anomaly detection using SVM and GA. In *Information Assurance Workshop, 2005. IAW'05. Proceedings from the Sixth Annual IEEE SMC*. IEEE, 176–183.

[28] Hossein Siadati, Bahador Saket, and Nasir Memon. 2016. Detecting malicious logins in enterprise networks using visualization. In *Visualization for Cyber Security (VizSec), 2016 IEEE Symposium on*. IEEE, 1–8.

[29] Jessica Silver-Greenberg, Matthew Goldstein, and Nicole Perlroth. 2014. JPMorgan Chase Hack Affects 76 Million Households. *New York Times* 2 (2014).

[30] Sara Sinclair, Sean W Smith, Stephanie Trudeau, M Eric Johnson, and Anthony Portera. 2007. Information risk in financial institutions: Field study and research roadmap. In *International Workshop on Enterprise Applications and Services in the Finance Industry*. Springer, 165–180.

[31] Robin Sommer and Vern Paxson. 2010. Outside the closed world: On using machine learning for network intrusion detection. In *2010 IEEE symposium on security and privacy*. IEEE, 305–316.

[32] Verizon RISK Team. 2017. 2017 Data Breach Investigations Report. (2017).

[33] Florian Tegeler, Xiaoming Fu, Giovanni Vigna, and Christopher Kruegel. 2012. Botfinder: Finding bots in network traffic without deep packet inspection. In *Proceedings of the 8th international conference on Emerging networking experiments and technologies*. ACM, 349–360.

[34] WSJ. 2014. Home Depot Hackers Exposed 53 Million Email Addresses. http://www.wsj.com/articles/home-depot-hackers-used-password-stolen-from-vendor-1415309282. (2014).

[35] Ting-Fang Yen, Alina Oprea, Kaan Onarlioglu, Todd Leetham, William Robertson, Ari Juels, and Engin Kirda. 2013. Beehive: Large-scale log analysis for detecting suspicious activity in enterprise networks. In *Proceedings of the 29th Annual Computer Security Applications Conference*. ACM, 199–208.