# Credential Theft Powered Unauthorized Login Detection through Spatial Augmentation

Zachary C. Burch

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Masters of Science
in
Computer Science & Application

Joseph G. Tront, Chair
Gang Wang
Aditya B. Prakash

September 28, 2018
Blacksburg, Virginia

Keywords: Security, Machine Learning, Login Classification, Spatial Augmentation

# Credential Theft Powered Unauthorized Login Detection through Spatial Augmentation

Zachary C. Burch

(ABSTRACT)

Credential theft is a network intrusion vector that subverts traditional defenses of a campus network, with a malicious login being the act of an attacker using those stolen credentials to access the target network. Historically, this approach is simple for an attacker to conduct and hard for a defender to detect. Alternative mitigation strategies require an in depth view of the network hosts, an untenable proposition in a campus network. We introduce a method of spatial augmentation of login events, creating a user and source IP trajectory for each event. These location mappings, built using user wireless activity and network state information, provide features needed for login classification. From this, we design and build a real time data collection, augmentation, and classification system for generating alerts on malicious events. With a relational database for data processing and a trained weighted random forests ensemble classifier, generated alerts are both timely and few enough to allow human analyst review of all generated events. We evaluate this design for three levels of attacker ability with a defined threat model. We evaluate our approach with a proof of concept system on weeks of live data collected from the Virginia Tech campus, under an IRB approved research protocol.

# Credential Theft Powered Unauthorized Login Detection through Spatial Augmentation

Zachary C. Burch

(GENERAL AUDIENCE ABSTRACT)

For a computer network, a common mode of access is a login; the entering of a valid username and password for authentication. Attackers use a variety of methods to steal user login credentials and several of these approaches are unnoticeable by network defenders. Providing further complications, a higher educational campus network, such as Virginia Tech, inherently has less information about the state of the network, since students and teachers bring their privately owned devices. To prevent this attack method, we determine the class, authorized or unauthorized, of login events using data that can be consistently provided by a campus network. After classification, alerts are generated for security analysts, helping to further defend the network. Spatial augmentation is a process we introduce to allow login classification with machine learning algorithms. For every login event at the campus, a history of user locations and source event locations can be provided, using data collected from the campus network infrastructure. Location data provides stronger classification of login events, since studies show attackers inherently have a physical distance between the normal user of an account when performing an unauthorized login. For evaluation, we build a system to augment and classify login events, while limiting the number of false alerts to a useable level.

# Acknowledgments

I would like to thank the members of my committee, Dr. Joseph Tront, Dr. Gang Wang, and Dr. Aditya Prakash, for their support and input in this research. Next, I also like to thank Randy Marchany and Dr. David Raymond for running the IT Security Lab and being available for on the spot advice. I would also like to thank Mark DeYoung for giving highly useful and practical advice, even if I sometimes didn't follow it, and providing excellent mentorship during his time at the lab.

Thank you to Phillip Kobezak for building the FINDIR system, which provided a strong foundation for building this work, and again to Mark DeYoung for the DAS dataset, which powered this research. I would like to thank everyone else at the ITSL who proviede feedback and assurance on this work, including Mike Cantrell, Brendan Mattina, Sam Hentschel, Ryan Kingrey, Samit Ganguli, Brad Tilly. I would espescially like to thank Jonathan DeFreeuw for helping me through everything at the end.

Next, I thank my family for their support and encouragement and time thoughout this, I would not be where I am today without them. Finally, and most importantly, I thank my loving fiance, Ginny, for her patience and love for this entire time. I do not want to think of where I would be without her.

# Contents

# List of Figures

# List of Tables

# List of Abbreviations

2FA Two factor authentication

IDS Intrusion detection system

NIDS Network intrusion detection system

# Chapter 1

# Introduction

Credential theft is a powerful attack method that subverts current means of intrusion detection for a campus network. When an attacker steals a username and password, they gain the ability to access target systems seemingly unhindered. User secrets can be, and are, stolen undetectably and used effectively for network compromise [1]. In 2017, 81% of hacking-related breaches were from stolen or weak passwords according to Verizon's annual report [2]. Even more recent phishing attacks by the "Manba Institute", an Iranian hacking group, support this claim with terabytes of stolen research, financial, and user data from more than three hundred universities around the world [3]. Login events are the key part of credential theft compromise. These intrusions require an attacker to login at least once with the stolen secret as the target user. If a login event could be classified as malicious or benign, then an entire class of intrusion vectors would be detectable [4].

## 1.1 Problem Statement

Two central challenges become apparent for login classification. First, limited information in a login event impedes complex learning approaches. Second, a proposed solution must be deployable for use by security analysts, using mechanisms for handling the fundamental challenges for anomaly detection in the field of network defense [5].

Network logging services provide a record of login time, user name, event source, and desti-

nation for every login event. User, source, and destinations identifiers are categorical and of limited use for classification purposes. While the data collection process itself is resilient to attacker manipulation, past attackers do manipulate both time and event source to mimic the benign activity of the targeted user [1], [6]. On their own, login record features are not effective indicators of compromise, irrespective of analysis method.

Certain constraints on any network intrusion detection system (NIDS) exist, independent of the method used [5]. A time-cost applies to any alert raised by a detection system, and defenders have a finite amount of time available for analysis. After a certain point these flagged events become ignored. False positive limitation is therefore a stringent requirement. For example, a "very tolerable" metric for a campus environment is 10 alerts in a day of 260,000 events [7]. A survey of past malicious login events shows that current detection of these attacks occur weeks to months after the event, compared to a desired time frame of seconds to minutes. [2]. A final restriction, bring your own device (BYOD) policies, common in campus networks, prevent collection of data directly from clients [8]. Devices on the network are not securable and only information obtainable from infrastructure is available. Viable solutions to login classification must address these issues.

## 1.2 Proposed Solution

We contribute the method of *spatial augmentation*, mapping user and event identifiers to geographic points, to provide needed discriminatory features for login events. We detail a model training approach to address past research limitations in the security fields. For model training and evaluation purposes, we use recent studies to develop a realistic attack synthesis mechanism, based on an in-depth understanding of the credential theft threat model. Empirical evaluation is done through the use of data from the Virginia Tech campus

| Name | Type |
|---|---|
| Latitude | Float |
| Longitude | Float |
| Time | DateTime |
| Accuracy (meters) | Integer |

Table 1.1: Data-type definition of a spatio-temporal point with accuracy (STA point)

and a proof-of-concept alerting system is designed and implemented. With this evaluation, we show ability to train accurate classification models for attack detection using spatial augmentation. In addition, our system design is build for streaming data processing and classification, providing alert generation capabilities seconds after an event occurs.

To provide an understanding of spatial augmentation, we first define a *spatio-temporal point with accuracy* (STA point) as the set of latitude, longitude, time, and accuracy values for a located identifier, shown in table 1.1. An *identifier trajectory* is a time ordered set of STA points. For login classification, these trajectories are always working back from the time of a given login event.

Table 1.2 lists the recorded features of a login and what information can be obtained by spatial augmentation. We refer to the owner of a network account as the *target user*. An *event source* is the source IP address in a login event. In specific, user location is found through logged interactions with a campus wireless network. Polled network state maps an internal IP to a campus room location, while outside IPs are geolocated through a third-party service.

Identifier trajectories are a set of latitude, longitude, time, and accuracy measurements with continuous values, such as integers or floats. These features are more suitable for machine learning based classifiers compared to highly categorical features such as IP address and usernames from the login event record. For model training, these values are scaled between

| Source | Name | Type |
|---|---|---|
| Recorded Data | Username | String |
| | IP Address | String |
| | Login Time | DateTime |
| | Destination | String |
| Spatial Augmentation | Target User Trajectory | [STA point] |
| | Event Source Trajectory | [STA point] |

Table 1.2: Basic and spatially augmented login event features.

0 and 1. Missing values are mapped to -1. For instance, a user who is never on campus will have empty trajectory values. Semi-categorical values are therefore produced, where the categorical aspect covers the missing values, and the continuous aspect are the normalized values between 0 and 1.

No other research efforts in the area of login classification are applicable in the constraints of a BYOD environment. Instead, enterprise network information from hosts are used, such as click events or internal domain authentications [7], [9]. In comparison, our spatial augmentation approach supplies location data solely from user and device interactions with the campus network. From that perspective, these interactions are robust to manipulation by an attacker operating under our defined threat model.

In building the spatial augmentation process, we extend previous works with network security data analytics to make a database system capable of streaming ingestion and search at the rate of incoming data. Challenges associated with were developing scalable methods and robust approaches. Through the streaming database, the spatial augmentation process was implemented in a parallel and efficient manner. For login classification we use a random forests model, based on the maturity of the approach, first introduced in 2001 [10], and it's past use in intrusion detection [11]. In addition, random forests can train from categorical and continuous values, making it suitable for our semi-categorical values produced by spatial

augmentation. Modeled attacker ability was varied from naive remote attacks to experienced actors controlling devices on campus

## 1.3 Organization

In Chapter 2 we formalize login events, then define our threat model and security environment. Background on weighted random forests and the threat model, campus environment, and noteworthy challenges are detailed. Recent works and the state of the art in detection systems related to the problem of login classification are discussed in Chapter 3. Spatial augmentation and the system designed to implement this process for the Virginia Tech campus is in Chapter 4. Chapter 5 provides a dataset analysis, streaming system ability, model evaluation, and empirical comparison to state of the art tools. The conclusions are in Chapter 6 along with the future work and limitations.

# Chapter 2

# Background

Our purpose in this chapter is to provide background on malicious logins and the campus environment. To start, we provide a definition for a login event, facilitating clear discussion. An unauthorized login threat model is detailed, followed by a description of a campus environment. Lastly, we list challenges associated with machine learning for usable network security. With the setting in place, we then describe core concepts for our detection scheme. Centralized logging services provide data flows for network analysis. We detail the classification model of weighted random forests and the properties of the model. A variant evaluation method is used, time-series nested cross validation, providing a robust measure of generalizability and prevents data leakage.

## 2.1 Login Event Definition

Multiple works in access classification provide a formal definition for a login event. We find that each definition has similar features, but differences and unstated assumptions create confusion. For clearer discussion, we detail and define the structure of a login and the associated record for it. We formulate these parts into three common types of account accesses: *service*, *machine*, and *network*. Service logins access an program running on a machine, machine logins provide access to a computer system, and network services provide ability to communicate with other connected systems.

### 2.1.1 Parts of a Login

From a high level, a login is the providing of information to an authentication mechanism, triggering an appropriate action. Specifics of each login part can vary depending upon the environment, level of security, and purpose. For reference, Figure 2.1 provides a tree based view of the conceptual pieces of a login. It provides a conceptual mapping of what information is in a login, what steps are part of authentication, and the results of a login.

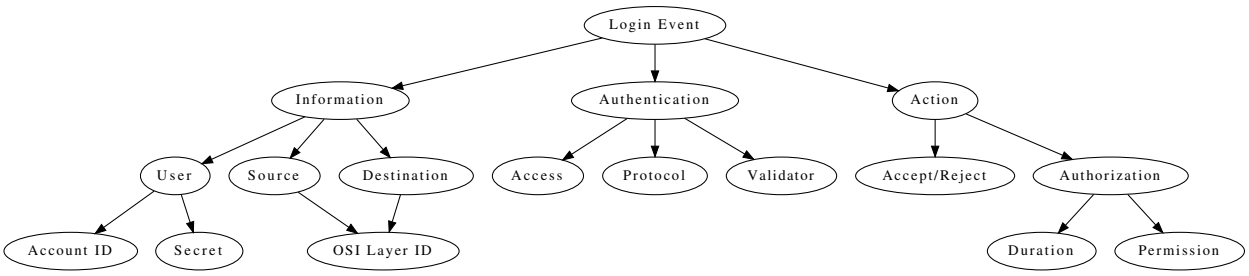


Figure 2.1: Breakdown of the parts of a login event.

Login information has user, source, and destination identifiers. A user identifier maps to a shared secret, usually a password, and an account ID or username. Both the source and destination map to an identifier on the OSI communication layer, such as an IP or MAC address [12].

Commonly, authorization and authentication are conflated. For example, an *auth* is completing a combination of the two ideas. However, there are excellent reasons to decouple the two concepts [13], [14]. *Authentication* is confirmation of a user's identity. Gaining permissions for a resource is *authorization*, and occurs after authentication. This division allows separation of powers between entities, providing security and flexibility.

Authentication mechanisms receive and validate user credentials against pre-shared information. Before a client can send a request, they must first initiate contact with the target system. Methods vary by login type. A multitude of protocols exist for secure data exchange,

examples include RADIUS, WPA2, and Kerberos [15]. The server's known user secret is then compared to the received secret to validate user identity. The result of that check determines the subsequent action.

Authorization varies by implementation, but some general features can be extracted from several standards [13], [14], [16]. For a valid authentication, permissions are granted to the source of the request. In the case of an invalid authentication attempt, restrictions can be applied, such as a timeout or firewall block. In either situation, these settings are only valid for a given period of time. Based on the type of login, differing permissions are granted. Service logins grant programmatic interaction, machine logins provide system usage, and network logins allow host communication.

A timestamp of when the event occurred is the final piece of identifying information. The definition's simplicity presents a problem of record keeping. What part of the event is 'when' it happened? We choose the latest point in time: when a user has completed authorization or failed in doing so. Other time points, such as the initial access or credential validation, were considered. Variance in protocols and implementations made using latest point in time the only consistent choice.

### 2.1.2 Login Types

From the definitions above, we formulate three main types of logins. The main differentiator is the target of the login, which affects permissions granted, authentication protocols available, and what information is transmitted. There are certainly edge cases which these descriptions do not cover. Still, these types cover login definitions seen in other works in this field.

*Service logins* occur when a user accesses a single limited resource located on a network or

host. Examples include logging into a web page, pushing data to a git repository, and accessing the printer service on a multi-user machine. Resources provided are not full machine functionality, instead usually only enough to fulfil a given task. From the perspective of the OSI layers, the source and destination identifiers are application level and above. We highlight a special case of service login, an authorization service login, where the capability provided by the service is authorization to a third-party service, machine, or network. A prime example of this is Shibboleth Single Sign-on[17]. Implemented at Virginia Tech, this authorizes users to websites, remote machines, and to the network VPN, which in effect encompasses all other login types.

When a *machine login* occurs, a user is provided control of the destination device. Locally, this is entering credentials into a screen prompt. Secure Shell (SSH) and Remote Desktop Protocol (RDP) are two common remote methods. Identifying information for remote accesses consists of the transport and network layers [18]. In contrast to service logins, the access itself has no inherent constraints on what actions can be taken on the machine.

A user gaining communication ability to other connected services and machines is called a *network login*. Source and destination data comes from and below the network layer. Protocols such as WPA2, OpenVPN and RADIUS, provide means of authentication and authorization. Commonly, some form of network login precedes either a service or machine login.

All of these login types can be centralized or decentralized, depending on implementation. That difference only affects unauthorized login classification by the amount of information that can be collected and contextualized. An edge case that might cause confusion is an authentication service login providing user authorization to access a machine or network. On first look, this appears to be a new type. However, this is internally multiple logins at one time. First of the user to the service, then the service to another target. This distinction

would not be possible without these definitions. In total, these types provide discussion clarity and cover most common logins.

## 2.2 Malicious Login Threat Model

When an attacker logins into a system using stolen credentials, we refer to that action as either a *malicious* or *unauthorized* login. A variety of methods for credential theft exist, but there is little variance in the use of stolen user secrets. This results in unauthorized logins against a target network [2]. In this section, we detail methods of credential theft, the malicious login threat model, and conclude with an incident summary of a recent attack at the Virginia Tech campus.

### 2.2.1 Credential theft

Phishing, data leaks, and malware are three main ways for an attacker to gain credentials [1]. Every approach provides the attacker with user passwords, the difference is in the method used. Phishing attacks are the most subtle, with no network indicators such as data exfiltration or command and control traffic. Instead, a normal seeming email is received, which deceives the user into inputting sensitive information [7].

If this attack method was localized, then it would be merely dangerous. But, users do not create unique passwords for every system and this attack becomes a widespread threat. In a recent study, [19] found that 80% of the study participants reused passwords, with on average 16% of those passwords being exactly reused per participant. Therefore, when an attacker steals from one system any other site with common users becomes a potential target. As a final hit, password partial reuse makes cross site awareness of this problem impossible, due

to hash comparison methods. In effect, defenders have no reasonable way of detecting the theft of user credentials.

### 2.2.2 Malicious Login

*Malicious* or *unauthorized* logins occur when an entity uses stolen credentials to access a target system. This definition handles a variety of attack methods, such as spear-phishing, brute force, password loss, keylogging, and more. Stated earlier, detecting the theft of credentials directly is not possible due to the efficacy of some methods. What is possible is the creation of a detailed threat model through data collected by past studies. This allows us to identify indicative features of a malicious login for later analysis [1], [6]. Building identifying features is an approach taken by similar works. Examples are credential based lateral movement [9], and spear-phishing [7].

Primary sources for this model creation come from two recent studies on malicious logins. Thomas, Moscicki, Margolis, *et al.* [1] analyzed how 1.9 billion stolen usernames and passwords had been used by adversaries. From this information, a geographic distribution of where phishing and keylogging attackers were was built. Onaolapo, Mariconti, and Stringhini [6] conducted a study of actions adversaries take post-intrusion and the pattern of their accesses when using a stolen credential. Most importantly, they published their data, providing more than 300 examples of malicious logins, with time and geolocation. Other works on password reuse [19], phishing attacks [20], and access controls [21] provide background on common methodologies.

Malicious logins separate into three sub-types based on the campus environment. We present these in order of least to greatest sophistication. *Remote* logins are any attacks with a source outside of an internal network. These attacks are the most common by far and

their distribution is known [1]. On-campus attacks originate from internal machines. *Static* on campus attacks arrive from devices with constant network identifiers, such as printers or servers. In contrast, *dynamic* attacks have no static network identifier, changing IP addresses constantly. Any devices connected to a wireless wide area network (WAN) fall into this category. Either of these attacks can occur from an adversary being physically on campus with their own device, or having compromised an insecure device. In total, we find that all of these variations have a consistent feature – adversary geographic location is different from target user geographic location.

In exception, an *impersonation* attack occur when a malicious login source originates from the device of the target user. More than just stolen credentials, this requires a targeted delivery of malware. Therefore, we consider this approach out of the class of malicious logins and out of our scope. Thankfully, this method is comparatively difficult and rare, making it less of a consideration to network defenders.

Irrespective of location of the attack, other features of attacker must be assumed for development. With the amount of variance shown in attacker methods, choosing any one feature would introduce bias. Therefore, we choose to consistently skew our assumptions towards a higher levels of sophistication. Ability to detect an advanced attacker implies ability to detect a basic one. We assume the following:

- The attacker has obtained proper credentials for the target user in a manner that is undetectable.
- No failed logins for the target user will occur as part of the compromise.
- Only one successful login will be needed for an attacker to reach their objective.
- The attacker has not taken over the target user's devices, so an impersonation attack will not occur.

- Services accessed are not indicative of the attacker.

We restate the features of the different types of malicious logins:

- Remote - Geolocation follows observed distribution [1].
- Static - Network location will not change over time
- Dynamic - Observable IP address is effectively random.

### 2.2.3 Example Incident - Subverting two factor authentication

The Virginia Tech campus has had previous incidents of credential theft. Because of this, two-factor authentication (2FA) has been implemented for all users of campus services. *Two-factor authentication* is the introduction of another identifying credential for authenticating the user. 2FA is thought to be an effective method of preventing credential theft, since while the first factor (username / password) might be easy to steal, the second factor is usually deemed much harder for the attacker to get [7], [22].

However, an incident happened recently where attackers successfully circumvented Virginia Tech's 2FA system. Through social engineering, the target user was manipulated into helping subvert the 2FA system, allowing a successful remote malicious login. This indicates how technical ability is not the sole judge of attack power and that 2FA is not a final solution to this problem.

In detail, a standard phishing scheme was used to convince the target user provide credentials to a malicious website, that imitated the general campus login page, login.vt.edu. Upon stealing the user credentials, attackers supplied them to the real campus login page, bypassing the first layer of security. Attackers were then confronted with the second layer of

authentication, where a choice of second factor method was provided. They chose the phone call option, causing a service to ring the target user's phone with a two-factor code. On the fake site, the standard phone call notification was shown, instructing the user to follow the instructions from the call. From the users perspective, the only suspicious indicator was that their choice of second factor was made for them. Therefore, they logged in correctly to the fake site. At which point, the attacker's gained the needed second factor information, and a successful unauthorized login had occurred.

## 2.3 Security Model

Generally, campus networks are defined by having a set of properties [23]:

- Set of buildings interconnected through multiple local area networks (LANs)
- Centralized network administration
- Defined geographic borders
- Permissive BYOD policies

These properties exist in both educational campuses, such as Virginia Tech or Stanford, and corporate campus networks, such as Microsoft [22], [23]. From a security standpoint, higher education and research create a need for supporting multiple complex use cases which might be restricted in a corporate setting. Administrative, educational, experimental, and residential needs must all be met in addition. All of these categories but administrative differ from an enterprise situation.

Educational needs necessitate flexibility for technologies that students or instructors might require, such as running new programs or bringing in equipment. Security researchers might

need to operate honeypots or malware sandboxes, which are highly restricted elsewhere. Connections to other institutions might cause large scale data transfers that look anomalous. Residents are streaming video and working on assignments.

All put together, the security operations at an educational campus must not get in the way of the users. Indeed, for the Virginia Tech campus, every type of activity mentioned is federated into it's own department, limiting workability by the defenders. Departments and individuals manage and operate their own equipment on the campus network. Summed up, oppressive firewall settings or communication restrictions are not feasible. Despite this, the onus of securing critical financial, personal, and research data is still on the security office.

## 2.4 Experimental Datasets

Services at Virginia Tech distribute information collected from or about the campus. Our work uses semi-static, polled, and state change data. Examples are respectively room coordinates, switch tables, and authentication events. We detail data collected for experimentation and the relationships those logs provide. Of specific note, personal identifiable information (PII) is part of this set. An IRB approved protocol, reproduced in Appendix A is used to de-identify the PII for research.

### 2.4.1 Data sources

Semi-static data is obtained from the campus facilities service, VT geographic information systems (GIS) research group, and the NetRecon service map. In this section, the data sources used in our work are detailed. *Semi-static* data refers to the case where collected information can be assumed correct, allowing for minor updates over a long period of time.

For example, building geographic location. Both *polled* facts and *state* events are dynamic data types, with fast rate of change. Network connections and logins are illustrative of this.

### Central Logging Service

The Central Logging Service (CLS) is a on-campus deployment of the Elastic stack collection platform [24] . Both polled and state change data stream log information to the CLS service, where it is received, buffered, normalized, and indexed for user queries. There are over 3,000 different log sources feeding into the CLS . It is a non-trivial fact that it is able to keep up with the incoming data in near-real time. A user can query and receive indexed answers from information that was received seconds ago. Some of the services feeding in are operating system events, application logs, and wireless and web user authentications.

Of particular note, another on campus service called NetRecon feeds it's data into the CLS. NetRecon is a polled data stream service that periodically scans all switches on the campus to log the state tables. This provides mappings between IP and MAC addresses and switch ports, roughly every five minutes. Three times a day, the associations of all switch ports and wall outlets are also logged. The polling approach means information collected can only ever be five minutes out of sync.

### Geographic Information

A geographic information system (GIS) is a database built specifically for spatial data. At Virginia Tech, an instance of the ArcGIS product is hosted. This provides semi-static data with building and room centroid locations. Also for location information, we use the public GeoLite2 dataset created by Max Mind. This company distributes variable accuracy mappings between IP addresses and world coordinates. The Virginia Tech GIS group maintains

data about all rooms on campus, with spatial mappings and shapefiles. NetRecon provides an updated list what switch outlets correspond to what room on campus. Combining these sources together, spatial information can be found for all device equipment on campus, and all IP addresses from off campus.

## 2.5 Challenges

Adoption of published research to commercial tooling is low for anomaly detection schemes. These challenges are detailed here, so they can be directly addressed in our designs. Sources of these challenges can generally be attributed to either the approach taken by past researchers or the specific properties of network intrusion detection.

### 2.5.1 Misclassification costs

False positives in an detection system incur a cost to the security analysts. This cost is in terms of both time and trust. Alerts require analyst time to process, contextualize, and evaluate. Time for other tasks is lost. Future events will not be treated with correct priority, if the system is seen to be error prone. [5]

Conversely, false negatives have a cost to the system, in that the detection scheme will classify malicious activity as benign. In the case of unauthorized logins, that malicious activity can have a high impact if the attacker is sufficiently advanced. In the case of Virginia Tech, incidents of this type have been highly damaging in the past, which caused the campus to adopt 2FA.

Put together, these two costs sets reveal a trade-off based on detector sensitivity. High alert count with reduce false negatives, while low alert counts will reduce false positives. Further

complicating this trade off, cost for each situation is dependent upon the services being accessed and the environment in question. A situation where the service in question has no security relevance might mean a much lower sensitivity is desired, while a high priority service with few users might demand a high sensitivity.

### 2.5.2 Data limitations

In the field of anomaly detection, there is a clear data imbalance problem. There is exists a corpus of benign data compared to a paltry amount of ground truth attack data. This one-sidedness provides an obstacle to many forms of algorithmic or machine learning approaches. Current approaches in other fields have a requirement of equal amounts of data in each class to train effective classifiers. An effective method in network intrusion detection must deal with this issue, either by using a method that is not affected by amount of data, or finding a way to get more anomaly data.

In specific, the KDD99 dataset, built for network intrusion detection research, has a multitude of flaws[5]. The data was generated on a simulated Air Force network in 1998, making the dataset nearly two decades old. Published research still uses this dataset, because it is easily available[11]. An extensive analysis done in 2003 by [25] found that certain artifacts made the dataset simple to overfit on. They provided a set of recommendations for counteracting these issues, but those appear to not have been followed.

### 2.5.3 Data variability

Login events are inherently tied to user activities. People have unique movement patterns, making each user a separate case. Even further, our actions can change significantly from previous days or weeks of activities. Vacations, sickness, and special events are all abnormal

behavior which change login patterns. An alerting system should be able to capture that distinction or be designed in such a way that it is agnostic to human variability.

### 2.5.4 Interpretable alerts

In past works, generated alerts by the proposed systems have been unreadable by analysts. Some forms of data normalization convert features in a manner that limits readability, since it varies based on the data and is not backwards translatable. Additionally, certain models, multilayer neural networks for instance, produce classifications that can not indicate the importance of features for a decision making process. Suppose a system correctly identifies a login as having a spatial violation, but can not indicate if the violation is due to the event location or the user location. The onus is then on the operator to find and connect the relevant details together for context. Only when that context has been created can the analyst make a decision about the alert. This cost is independent of the sensitivity costs. To solve this problem, an alerting system must provide context.

### 2.5.5 Intelligent adversaries

Culminating the set of challenges to network intrusion detection, is the simple fact that attackers are smart. A model that detects adversaries for day one might not work for day two. Advanced persistent threats are non-financially motivated. This means large amounts of time and effort can be devoted to find means of avoiding detection. Therefore, it is imperative for these systems to be built in such a manner that acknowledges the primary method of alert avoidance, feature manipulation. Information the system uses to make a decision must be resistant to attacker manipulation, otherwise there is a straightforward means of avoidance.

## 2.6 Weighted Random Forests

Random forests is a learning algorithm that is both resistant to overfitting and works with continuous and categorical data [10]. Overfitting occurs when a model doesn't properly generalize from training data to testing data, and highly impedes performance. A specific variation, weighted random forests, provides protection against bias from class imbalance [26]. Before explaining the random forests algorithm, we first provide background on their component pieces – decision trees.

A decision tree is a tree of branching conditions that differentiate data. Each interior node encodes a function which determines the next branch. On reaching a leaf node, a final classification is made based on prior probabilities. They are trained, or 'grown', by building rules off of labeled data. Several competing growth methods exist (CART, C4.5, and QUEST) [11]. A valuable aspect is they are structurally straightforward for a human to interpret, compared to other classifiers such as neural networks [27].

To illustrate this concept, consider Figure 2.2. A tree is trained for predicting if children will be playing outside, based on weather data. For instance, take a day with a sunny outlook, no wind, and a high temperature of 92.3. From the root node, we make a decision on based on the outlook, then on the temperature. After the first decision, we could have decided YES with 77% (7/9) certainty. But, the hot day causes us to continue and finally choose NO with 66% (2/3) confidence.

Random forests are sets of $k$ decision trees, each grown off of an independent random vector. A random forest works by taking the majority vote for assigning a given sample to a class. Commonly, $k$ is set to 100. This $k$ value provides a simple percentage value of classification confidence for a random forest where each tree acts as one percent. This approach is called *ensemble classification*. Overall, random forests provide the following benefits [10], [27], [28]:

- Inherits understandability of decision trees.
- As $k$ increases so does generalization due to Strong Law of Large Numbers.
- Ensemble approach provides robustness against outliers and noise
- Classifier training is easily parallelized to $k$ threads.

All of these aspects provide clear strength that are applicable to the security domain. We must also clarify a few weaknesses of random forests. First, they are non-parametric. As training data and features grow so does computational cost. Next, random forests require extensive feature engineering compared to deep neural networks [29]. We leverage our domain expertise to make effective features for the classifier, and find empirically that current computational power is sufficient to use random forests.

The most important weakness is random forests are just as susceptible to class imbalance as other classifiers. To handle this issue, we utilize a variation named *weighted random forests* [26]. Normally, a random forest would overfits the majority class when training. To handle class overfitting, cost-sensitive learning is used to assign a higher penalty to misclassification of the minority case. Therefore, the model can be tuned to achieve the best performance for the desired class.

## 2.7 Time-series Nested Cross Validation

Training methods are always an important consideration when using learning algorithms. For the security domain in specific, the important consideration is generalizing from the training data to real attacks. Effective training methods help prevent future false positives and reduces the analyst costs [5]. To provide both of these features, we use a variation training

methodology called *time-series nested cross validation*, a training method that prevents data leakage when handling temporal data.

A standard technique, *k-fold cross validation* ($k$-CV) is the practice of splitting data and combining the $k$ segments into sets of training and validation data. For each set, model training and hyper-parameter tuning occurs. Model evaluation is done, providing $k$ error values. Averaging the error values provides an overall error value. This training method provides a strong measure of model performance and allows robust tuning of hyper-parameters [27].

In time series data, $k$-CV can introduce a form of bias called data leakage [30]. For instance, consider if a model was trained on a segment split temporally from $[t_2, t_3)$ and a segment from $[t_1, t_2)$ used as a test set. The later segment is dependent on the earlier data, providing the model with information of the test set during training. An algorithm trained with that data would be interpolating data instead of predicting.

To prevent data leakage, we must implement a modified method called time-series nested cross validation. The difference is in how train/test pairs are created. $k$-CV provides sets consisting of all possible variations for a data set. Time-series nested is the process of building pairs that maintain temporal order over time. This was introduced by Cochrane [31] and has the effect of mimicking training on semi-real time data and preventing data leakage.

Time-series nesting is the process of building the needed train/test pairs for CV. Each pair is "nested" inside the subsequent pair, extending in a temporal fashion. Consider a set of $n$ data segments, where each segment is sized temporally, such as blocks of hours, days, or months. A training set includes the first $i$ blocks, where $2 \leq i \leq n-1$. The $i+1$ block is the test set. Since a training set needs two subsets, training and validation, and one period is needed for final testing, the minimum size is three periods for one evaluation phase. Each

test set and validation subset is one block in size, while training subset size varies. Therefore, a maximum of $n - 2$ pairs can be made. Figure 2.3 provides illustration of this concept. Similar to $k$-CV, we calculate the total error by averaging error of each set.

## 2.8 Chapter Summary

In this chapter, we show unauthorized logins are a immediate and effective threat to networks. Even worse, a campus network's defense model has additional limitations that further reduce the powers of the analysts and potentially increase the attack effect. We detail the data sets we have collected from Virginia Tech. Due to the sensitive nature, we follow an IRB approved protocol and work with the campus' security office to de-identify the information, allowing ethical research. Finally, we detail critical design challenges in this field. These challenges provide a basis of comparison between our work and other related works. A guarantee of generalizability and usefulness is also provided by the structure of weighted random forests and the time-series nested cross validation approach.

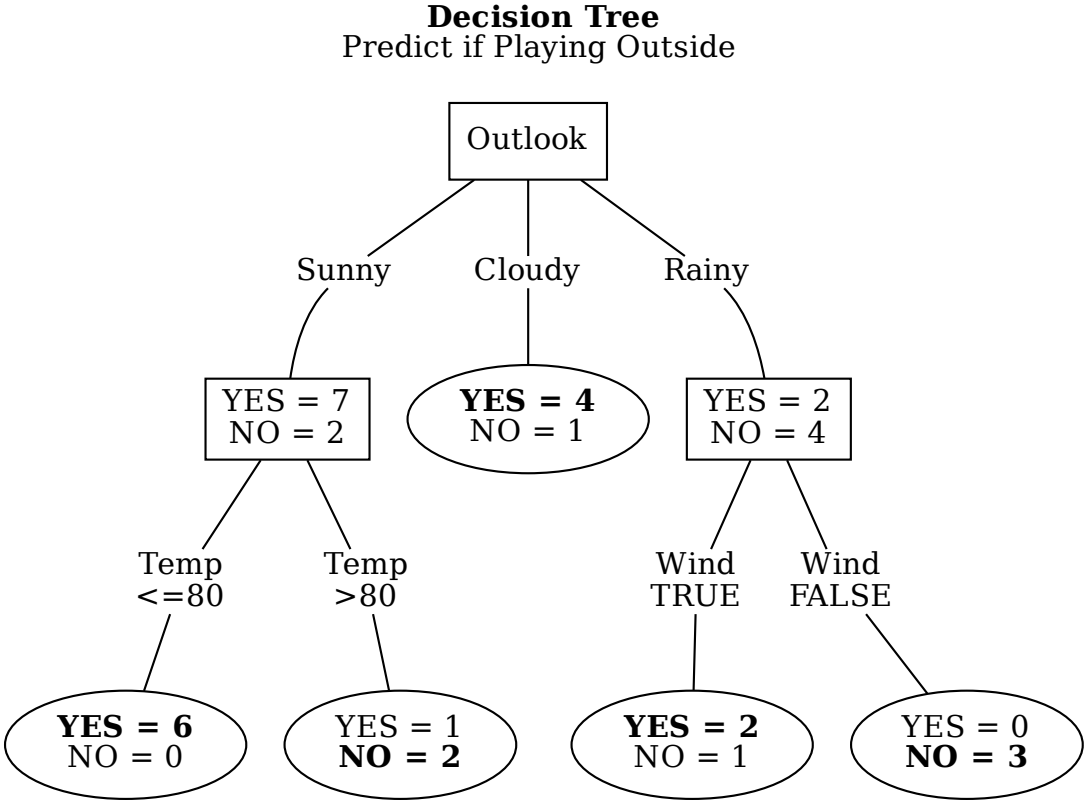


Figure 2.2: Example decision tree that uses weather features to determine the probability someone will be playing outside later. Training data is a set of days where kids did or did not play outside for a set of weather conditions. Both continuous and categorical variables are used. Final predictions have a confidence based on previously seen events.

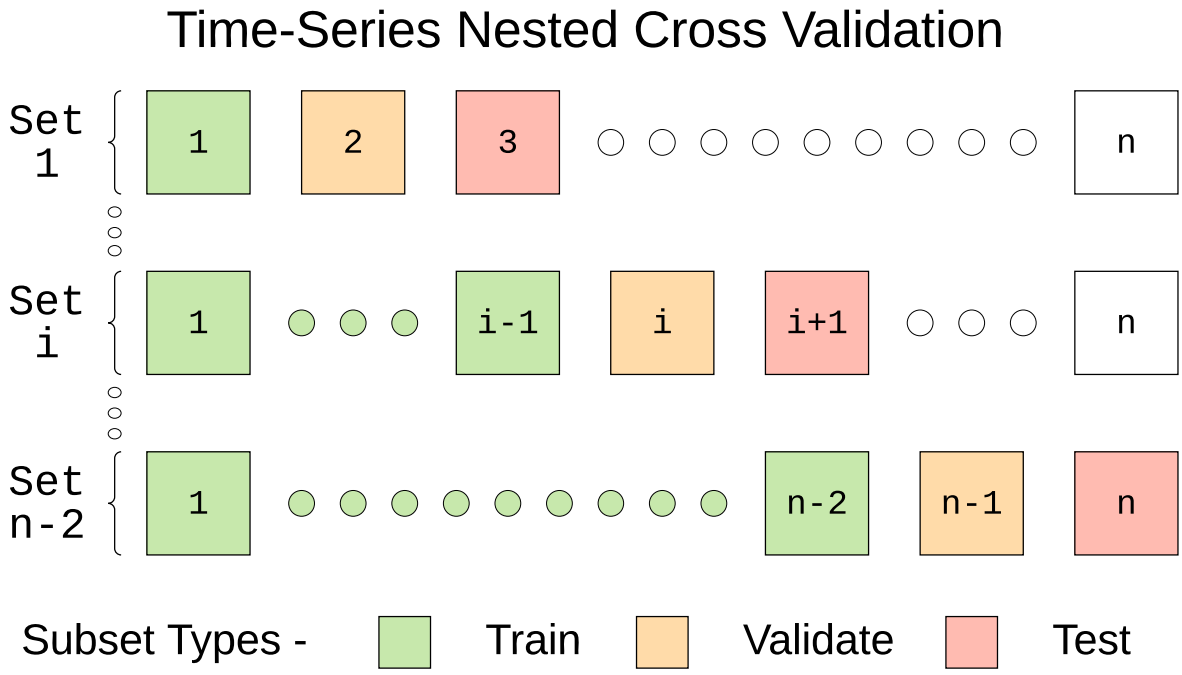


Figure 2.3: Illustration of time-series nested set creation for cross validation. Each block represents data from a period of time, such as an hour, day, or week. A total of $n-2$ sets are possible from $n$ blocks. After a model has gone through tuning for the training and validation subsets, error is calculated on the test subset. Total model error is an average of each set error.

# Chapter 3

# Previous Solutions

Analysis of login events is not a new field, but no serious studies had been conducted in login classificaiton until 2010. Since then, only until recently it has not been well explored. Another indicator is the non-existence of commercial tools for sign on classification. For all these related works, we make clear if and how geolocation is used.

In this chapter, we first discuss the state of the art for access event analysis tools. These programs have limited detection capabilities (i.e. brute force attempts). Otherwise, they provide alternatives such as analyst assistance for manual analysis and mitigation strategies through access control. In two cases, systems provide comparative location

Then, we detail other research efforts that try to address this problem in other environments. In the past few years, multiple new studies have been published presenting different schemes for malicious access detection. But, these approaches are fundamentally infeasible for a campus environment.

## 3.1 Available Tools

There are a multitude of network and host intrusion detection systems commercially and freely available. Unauthorized logins are clearly a form of intrusion. But, the nature of the attack subverts many of the current alerting models. Overall, this has lead to systems

working to assist the analyst or provide mitigation strategies. Even so, a traditional IDS can prevent simple brute force attacks. Contrasting the general approach is the recent GeoLogonalyser, which follows our work by alerting directly on login events.

### 3.1.1 Brute Force Defense

One clear need current systems fulfill is the detection and mitigation of brute force password guessing attacks. Multiple tools, commercial and free, exist that meet this need. An excellent example is the tool "Fail2ban". Using access logs, it detects if a given client matches a given malicious pattern, then automatically responds with another given action. An option to use IP address geolocation information for blocking addresses exists. But, it only allows blocking static geographic areas, with no other room for expansion. [32]

### 3.1.2 Analyst Assistance

Even though automatic classification of logins are difficult, manual analysis has power. Existing system's recognize the goal of empowering the analyst. One method is advanced report generation on the various login activities, used in multiple tools [33]–[35]. An example of this is displaying all instances where a user had logged in from a different IP address or machine (depending on the environment).

Assisting the analyst is a small, but definite method of improving defensive power. What this information does miss though is the ability to provide a history that might be of use to the analyst. From what is surveyed here, location is only displayed to analyst, though some tools to allow filtering by city or country.

### 3.1.3 Previous Location Alerting

After extensive searching, we have identified two commercial systems that do allow leverage geolocation for analysis. This occurred through internal displays or modifiable plugins. We were able to get the source code for both of these tools, which allowed an in depth analysis of the approach.

Splunk, a log analysis system, has a *User Activity Monitoring* tool which provides a *Geographically Improbable Access* view. On the back end, the view is 12 line query. The approach is to geolocate the current and previous logins for users, then calculate a miles per hour speed between the two. If that speed exceeds a hard coded value of 500 mph (224 m/s), then the suspicious login event is shown[35].

*GeoLogonalyser* is a utility provided by FireEye, a popular NIDS. The goal of the project is to identify anomalies in remote access VPN logs. The set of target anomalies are login rate, travel speed, and value changes in autonomous system number (ASN), hostname, or VPN client. It dynamically updates itself using MaxMind GeoIP2 databases [36], which provide semi-accurate mappings between IP addresses and geolocation. Since the code for the tool was available online, written in Python, we dug into the alerting mechanisms. There were two location based alerts, FAST and DISTANCE. If the difference between two logins was over 500 mph or 500 miles, the anomalies were respectively raised. City and country information was also displayed for the analyst, but not used for anomaly detection.

We note that both of these tools compare the current login event to the previous one. They use the IP address to get the location of the event, which is vulnerable to attacker manipulation [6]. Additionally, GeoIP is not useful for a network with internal addresses. Beyond that, the threshold values, which perfectly reasonable, lack granularity. What might be a reasonable speed over an hour might be perfectly impossible in ten minutes.

### 3.1.4 Access Control

Despite the focus on unauthorized login **detection**, mitigation strategies exist in the form of access control methods. Role based access control (RBAC) is commonly used in industry networks. Permissions are tied to simple roles, which are then assigned to authorized users. From a theoretical standpoint, this is an excellent approach. Attackers who gain access to a non-critical user can only effect limited amounts of harm. In real world cases though, this model is difficult to support, with users continually being over provisioned with permissions (50% - 90%)[21].

Further marginalizing access controls, they are not effective for a campus security model. First, the controls reduce usability, which is critical for campus operations, detailed in Section 2.5. Second, these controls do not protect critical resources from the perspective of a user, since an attack who gains access to an account still has power of them. Finally, an attacker who gains access to one account on a network has tremendous power to pivot farther upward in control through social engineering attacks such as spear-phishing from a valid email address.

## 3.2 Research Efforts

In this section, we analyze other works in the field of malicious access detection. Proposed methods range from visualization to pattern mining to bespoke signatures. We discuss specifically how these works approach the challenges detailed in section 2.5. Most fail to achieve a realistic number of daily alerts, and instead throw thousands of false positive. Approaches successful in that category are not applicable to a campus environment.

### 3.2.1 Access Patterns

Patterns in malicious accesses commonly differ compared to those of benign users. Siadati, Saket, and Memon [4] developed APT-Hunter1, a visualization tool, to empower analysts to investigate enterprise login patterns. Analysts are presented a graph of networked machines, where handwritten filters can be interactively applied. Evaluation was done through an informal user study of two analysts exploring the LANL dataset [37]. 349 out of 749 malicious logins were identified in a "few hours" for 30 days of data, with a daily average of 549 false positives.

Based on their previous work with access patterns, the next year Siadati and Memon [9] published further work. Keeping the initial idea, they implemented a pattern mining approach for alerting. Here, they formally defined a login as a tuple of user, source machine, and destination machine. For their evaluation, data from an unnamed financial company provided a baseline, and synthetic attacks were inserted. Reported results showed 82% true positive rate, with only a 0.3% false positives.

Overall, these are two excellent works. Pattern mining is a powerful tool and is implemented well. Unfortunately, they are fundamentally incompatible with a campus environment. Only service machines provide access information to central analysts, clients do not. Additionally, due to the dynamic nature of network addresses, reliable source identification cannot be achieved in the campus environment, compared to the enterprise network. One final note, even though a false positive rate of 0.3% was achieved, that still translates to 480 false alerts per day.

### 3.2.2 User Fingerprinting

Another commonality of malicious logins is the difference between attacker and user features. M Freeman, Jain, Duermuth, *et al.* [38] explored this through a process called *reinforced authentication*. The key idea is using complementary information to further validate login events. Cookies, user agent string, and IP geolocation were proposed to be used in conjunction with the user's history to determine access validity. In their paper's implementation, they only used user agent string and country location as features. At an untenable 10% false positive rate they were able to achieve a 77% true positive rate.

Tangentially to the previous work, in 2011, [39] wrote about using user behavior as implicit authentication. The purpose was less attack prevention and more focused on providing better usability of devices. Proposed was the thought that, since people always carry their smartphone with them, collected data from sensors could provide proof to other services about the validity of the user. Fingerprinting was done through phone GPS, browsing history, phone calls, and text messages. Those features together were able to uniquely identify the user based on the testing they were able to accomplish.

In summary, both of these works made the case that an attacker could not replicate the user well enough to fool the detection capabilities. But, more recent works have shown that assumption to be definably false. Thomas, Moscicki, Margolis, *et al.* [1] found that over 80% of phishing and malware approaches steal not just user credentials, but details like geolocation and full name. Onaolapo, Mariconti, and Stringhini [6] showed that when attackers are able to get geolocation for login credentials, there is a statistical difference in where the access come from. Adversaries are smart and are able fool or avoid these attack with apparent ease. A final note, the campus environment prevents approaches such as getting information directly from user devices, making finger printing much harder.

### 3.2.3 Credential Theft Prevention

Grant Ho, Aashish Sharma, Mobin Javed, *et al.* [7] published an excellent work in 2017 about how to detect credential theft. The goal was to prevent credential theft, but did some work tangentially on detecting unauthorized logins. They broke down credential theft based phishing emails into three categories, one of which dealt with malicious access. A lateral attacker is when an user email account is used to send more phishing emails within the company. Ho et al.'s detection approach was to geolocate potentially malicious internal emails and then extract two features. Namely, number of users in the company from that city and number of times that specific user sent emails from that city. These were then used in training a simple detector. Their evaluation used real email and attack data from Lawrence Berkley National Lab, and found 90% of labeled attacks, with only 10 alerts produced per day. Additionally, they found two previously unknown attacks using their approach.

Even still, this technique doesn't work for the campus environment. First, logged click events are used to identify potentially malicious emails, which cannot be obtained without client monitoring. Second, their technique works well with a widely diverse organization, but would be less effective with a centralized campus. Their historical information could be replaced with a simple rule to alert on all logins from off campus. Concluding this, [1] found that the majority of phishing attacks collected geolocation from target users. All an attacker would need to do to avoid this situation is use a VPN service to get proper city locations. This attacker approach is shown in [6].

### 3.2.4 Summary

We established the state of the art and related works that are relevant. In doing so, we analyzed their approaches and compared them to their applicability to a campus environment.

Overall, solutions used relied on being able to get trustworthy information about the user in question, which is not feasible outside the enterprise network. A viable solution must be able to work within the main restriction of only collecting network event information.

# Chapter 4

# Detector Design

We begin with the set of design goals for our work, based on the limitations of other works and the challenges of the field of login classification. From there, we describe the process of spatial augmentation from a high level view, which is appropriate for networks similar to Virginia Tech's. With that concept, we design and implement a real-time streaming relational database in TimescaleDB that is appropriate for receiving of network event data and storing geographical mapping for loading and login augmentation. To provide labeled model training data, we design a method of synthetically generating malicious logins for our target campus, in a manner that avoids potential sample bias. After providing a stream of labeled augmented login events, we use the collected data for training a weighted random forests model. With a base model in place, we use that model to classify new events and generate interpretable alerts. Shown in Figure 4.1, our design falls into four phases of operation: data processing, spatial augmentation, login event classification, and nightly model training.

For this chapter, we start with the core concept of spatial augmentation, then traverse back along the data flow into data processing and the implementation in a time-series relational database. Moving forward, augmentation generates data for a weighted random forests classification model. To train that model, we must create labeled attack data, free of bias, using studies on malicious logins to synthesize training events.

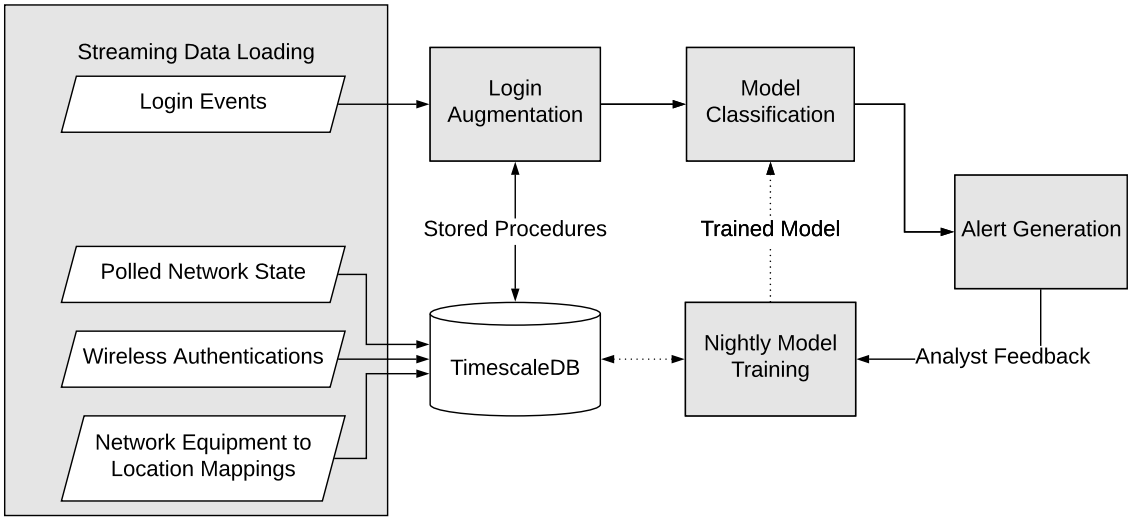


Figure 4.1: Detection System Design.

## 4.1 Design Goals

Overall, our main goal is to research and develop a process of detecting malicious login events with spatial information in a readily deployable manner. We take that high level view and separate out a set of design goals for our detector. Of course, the first, and most important goal, is to accurately classify malicious login events. Beyond classification ability, our detector design fulfills the following set of objectives for a deployable system. These goals are distilled from Chapters 2 and 3. After stating our goals, we provide the rationale behind them.

- **Campus Login Event Classification** A login event from a campus environment needs to be classifiable as malicious or benign.
- **False Positive Minimization** Amount of false alerts generated by the system should be minimal, on the magnitude of 10 per day.
- **Real-time Alert Generation** Malicious events should be alerted to in as close to

real-time as possible.

- **Design Generalization** A system design is flexible enough to account for new attack approaches and keep current with data seasonality and drift.
- **Alert Interpretability** Alerts have to contain features explaining *why* it was generated.
- **Low Hardware Requirements** Processing, storage, and networking requirements fit within a campus environment's limitations.
- **Mature Technology** No commercial tools, proprietary algorithms will be used. Preference is given to mature technologies.

Most importantly, our purpose is to design and test a method of real-time login event classification, therefore the first goal is successful classification of login events. This is a vague requirement since "successful" is not defined. This leads to minimization of false positives, with a concrete goal of generating less than 10 alerts per day if possible and a hard limit of 50 per day. We choose this metric based on a related work, spear-phishing alerting, for a similarly sized network [7] and on our knowledge of the security analysts at Virginia Tech. Completing our focus on the performance, a working model should be able to grow to detect new attacks and account for long-term changes in a network setup. Models with highly specific features perform well for a given dataset, but new information sources break existing functionality [5].

Real-time requirements for alert generation are based on past events of credential theft which have had response times in the range of months to years [2]. We determine that a useful system should provide the ability for an analyst to respond or a restriction to be applied. Keeping with the vein of usability, an alert should be easily human comprehensible. A more

comprehensible alert reduces analyst read time. Interpretability has been a needed factor in some recent works using a machine learning based approach [5], [11].

Moving past analyst time cost, the three goals of low hardware requirements, mature technology, and design simplicity are all based on core concepts of a deployable system. First, many machine learning methods require expensive GPUs for training, upping cost of entry. Second, some software tooling for network logging and data processing is highly priced. We instead choose to use open sourced systems and libraries to avoid barriers to entry. Third, a mature technology, in the open source sense, is one with an active community and history of successful deployments. While each of these goals separately address a particular, distinct issue, they collectively work together to assure a detection system that is built to be used.

## 4.2 Spatial Augmentation of Login Events

Spatial augmentation is the mapping of dynamic network identifiers to physical locations. In spatial augmentation for login events, we augment the user identifier and the source. These contextual features of login events provide a view into the known spatial anomaly that exists when malicious logins occur.

Latitude and longitude values for both the username and source IP are the result features for every login event. By both definition in our background and experimentation by [6], it is known that malicious logins have a spatial anomaly between the target user and source. Therefore, these values can provide the needed context for classification, with a few assumptions:

1. Information must be resilient to attacker manipulation.

2. Measurement precision should be high enough to reveal anomalies.

3. Since human movements affect these values, noise and inconsistency must be accounted for.

4. Benign edge cases must be either separable from malicious events or filtered out in some manner.

To obtain the required data, network logging services provide the needed dynamic state information. Spatial mappings for a campus commonly exist and if not, they can be built from combinations of inventory to room mappings and room to GPS mappings. By joining these data together, for a given point in time, a network identifier can be mapped to GPS coordinates with a known accuracy.

Identifiers of importance to spatial augmentation for login classification are user identifier and login source IP address. We do not consider target application to be important for classification, since critical campus services are in common use, such as email and finances. To augment a user ID, enterprise and educational wireless authentication log data maps a given user's devices to on campus access points. Joining this log information with access point to GPS mappings, we obtain user location on every wireless access. Two cases exist for IP addresses: internal or external to the campus. External, or off-campus, IP addresses are mapped to location using a third-party service, such as MaxMind [36]. Internal IP addresses are mapped to network equipment using polled information about switch states. Again, room Ethernet ports are mappable to GPS coordinates using prior knowledge of the campus environment. DHCP addresses are also included in this mapping, since they are communicated over an access point or router's MAC address. All of these location mappings have an accuracy value, accounting for ranges of wireless signals for internal events and uncertainty for external events. Figure 4.2 provides a visual flow of this process.

Measurement precision can interfere with classification ability. There are two sources of

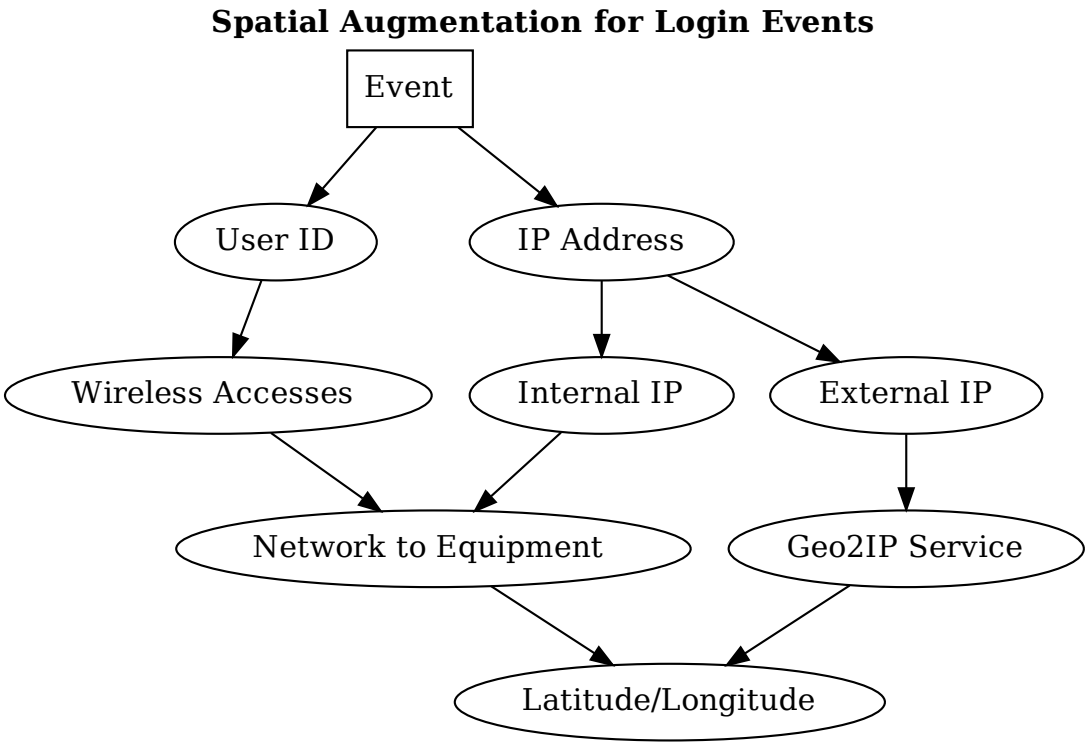


Figure 4.2: Spatial Augmentation Process

variance. Spatial accuracy is the impreciseness in geolocation. Temporal offset occurs when the time of a user location differs from the time of a IP address location. We include these features along with geographic values for a model take into account for classification. Table 4.1 provides features per login, with the known ranges.

Considering other potential issues for the spatial augmentation process, it becomes apparent that malicious events violate a pattern of user behavior compared to benign edge cases or human movement patterns. To capture that pattern for algorithmic analysis, we add in historical user location and historical source location for a current user login. This provides the model with the last $n$ actions a user took and the activity of a given event source,

| Source | Feature | Range |
|---|---|---|
| Event | Latitude | [-180.0, 180.0] |
| | Longitude | [-90.0, 90.0] |
| | Accuracy (internal) | [1, 120] |
| | Accuracy (external) | [1000, 1000000] |
| User | Latitude | [-180.0, 180.0] |
| | Longitude | [-90.0, 90.0] |
| | Accuracy | [35, 120] |
| Time | Difference | $[0, \infty)$ |

Table 4.1: Features generated by the spatial augmentation process for a Shibboleth login event. Accuracy values are in meters, latitude/longitude in degrees, and time in seconds.

providing additional context, which is still trustworthy and human understandable.

Trustworthiness of this process is based on the control of incoming data sources. By definition of a campus network, infrastructure equipment is centrally managed by the same organization that provides security. This direct device control confers a high amount of trust. Imitating this information would require some form of network equipment compromise, which is out of scope for our threat model. Previously stated, information that is manipulatable by an attacker are the geolocation of logins from an external IP. But, user ID location is not affected by this approach, and spatial discrepancy becomes apparent.

We can now view client devices based on how they interact with the campus network and how they move around geographically. A rich amount of data can be extracted on the devices, and their associated user account. Compared to enterprise setups, where logging services are installed on all network host, we obtain slightly less data for a much lower maintenance cost.

## 4.3 Time-series Relational Database

For both the data collection and spatial augmentation process, a database is needed that can perform the necessary functions in a timely manner. Two types of relationships need to be stored and searched: static and time-series. This presents a design challenge based on current technologies. Relational databases, or SQL databases, are built for searching across sets of relationships, but not optimal for streaming data. NoSQL databases are the converse, providing strong loading and general search capabilities, but fundamentally incapable of the relationship mapping provided by SQL systems. Based on these trade-offs, we choose to implement our procedures in a SQL database system. While SQL-based systems might not be optimal for time-series data, there is no fundamental limitation to their extension. More specifically, we use PostgreSQL 10, which is an open sourced relational database that has been developed for over 30 years [40], meeting our goal of using mature technologies. An example of this is how PostgreSQL has built in types for efficiently and natively handling comparison and searches with IP address, MAC address, and time range values. Comparatively, some newer technologies have not had the time to implement such functionality, meaning additional effort for our work.

An immediate benefit of choosing PostgreSQL is the existence of an extension called TimescaleDB [41]. TimescaleDB provides a set of additions to PostgreSQL that are optimized for time-series data. This is done with the addition of the concept of hyper-tables, which are a super set of smaller tables partitioned by time. In our design, we set our partition size to be one day. Each day of data is loaded in its own table. In PostgreSQL, a table index scales with the number of inserted rows. Splitting by day prevents large growth, since each day has its own index. This does not come without cost. Public keys are limited, since they must include the table's time field as an element. Non-time based searches suffer, because each day's index must be checked, but time-ranged based queries are more efficient since only the

tables with the needed information are included in the search.

With fundamental technologies decided upon, we describe the features of information stored in our database. After that, we explain our reasoning behind our table and index connections. Insertion and augmentation functions, stored inside the database, are built in a manner that is thread safe and streaming capable. We conclude our details with optimizations used to make the database performant.

### 4.3.1 Streaming and Static Information

Before discussing table structure, we detail the type types of data that the database stores. Static data is information such as GPS locations for campus rooms. This information does not change over short periods of time, but instead over months and years. In contrast, dynamic streaming data always has a time stamp attached to the information recorded, since it is only guaranteed to be accurate for a small window of time. A good example is an IP address assignment, which only applies for a set period after it has been assigned. Static information can be preloaded into our database, meaning batch insertion methods and occasional updates can be applied outside the streaming process.

Four types of static mappings are collected and used to provide the capabilities for spatial augmentation. Three are from internal campus sources, with the last being external. Starting with the external data, Geo2IP is the mapping between IP address ranges and physical locations. Our source of these data is a company called MaxMind [36]. Internally, NetRecon provides a static source in the network switch port to Ethernet wall outlet mappings and each wall outlet has an associated internal space ID. Similarly, through the campus facilities service, we obtain access point name to room space IDs. Finally, the campus GIS group provides a mapping of space IDs to GPS coordinates. In total, this means every registered

| Data Type | Name | Features |
|---|---|---|
| Static | Geo2IP | IP Address Range, Latitude, Longitude, Accuracy |
| | Port to Room Outlet | Switch Port, Space ID |
| | Access Point to Room | Access Point Device Name, Space ID |
| | Room to GPS | Space ID, Latitude, Longitude |
| Dynamic | IPv4 to MAC | Time, IPv4 Address, Mac Address |
| | IPv6 to MAC | Time, IPv6 Address, Mac Address |
| | Mac to Port | Time, Mac Address, Switch Port |
| | Wireless Authentications | Time, User ID, Access Point Name |
| | Single Sign On Logins | Time, User ID, IP Address, Application |

Table 4.2: Information on the types of data collected

access point and Ethernet room outlet map to highly accurate internal campus locations, while outside IP addresses map to low accuracy GPS locations. All of these sources are collected prior to testing and preloaded into our database.

Static data sources do change somewhat over time for any location. Rooms might be remodeled, switches replaced, and new buildings built. This incurs a maintenance cost upon the system of upholding network information. However, many companies and universities already maintain an equipment inventory and have site maps, which can offload this cost to other organizational services. In the case of Virginia Tech, three services, NetRecon, facilities, and the GIS group, provided pieces of the needed information.

Dynamic data is obtained from three logging sources: NetRecon, Aruba Station Manager, and Shibboleth Single Sign On. NetRecon provides information about the observed connections between IP addresses and MAC addresses viewed over network switches. Aruba Station Manager is the name of the service which manages authentications for the campus wireless network. Last, Shibboleth SSO acts as the campus authorization and provides the login data for the network. Note, Virginia Tech has a separate authentication and authorization service for the wireless LAN compared to SSO. All of these services provide continuous logging information for either a change in state or every polling period. NetRecon polls the

campus switches every set period, which is 5 minutes for the data we are using. In contrast, the other two services provide data every time an event occurs, specifically a successful login.

All of this data is used for spatial augmentation and is carefully stored in our database, using table and index layout described in the next section.

### 4.3.2 Table Layout

For spatial augmentation, we layout and implement a tables structure for the data collected, keeping our design goals. We note here that we build off of the table design from [42], though we extend the spatial capabilities and limit the stored network data for our goals of streaming spatial augmentation. Additionally, we extend our capabilities through TimescaleDB and parallel functionality. For reference, table 4.3 provides a listing of each table's name and the stored relationships withing the tables.

Starting with the basic tables that map identifiers to values, the application and authenticated user tables provide mappings between a unique integer ID and full information. In the case of the applications, it provides the full URL used in the SSO process, which is of value to an analyst from an interpretability standpoint. For the authenticated user, the ID maps to a user name and domain. Lastly, the campus location table provides a mapping of space ID values to latitude/longitude coordinates for on campus locations. All of these mappings are one-to-one and unique. They exist to limit the amount of information that has to be replicated in the database. For instance, the application table allows an integer to be stored in place of a long URL string.

Our next group of tables are many-to-one mappings: IP type, network equipment ID to space ID, and GeoIPNetwork. IP type is a listing of the campus IP ranges and their associated properties and meaning. Many IP addresses map to a single description in this table. Both

IPv4 and IPv6 address ranges are included in this mapping. The next mapping is the one obtained by a combination of the data from NetRecon port to outlet and campus facilities. It provides the mapping between network equipment IDs to campus location space IDs. One space ID, or room, can have a mix of access points and internet outlets if big enough, making this a many to one relationship. Finally, GeoIPNetwork mappings are provided by the MaxMind service [36]. Again, several IP addresses from different ranges can map to the same geographic coordinates, especially with the comparatively limited accuracy.

Hyper-tables are the final group of tables, and the most complex. Four are used, each one for a different data stream. Each hyper-table is configured to store every one day interval of data in a sub-table. Every sub-table keeps its own indices for the records within. All hyper-tables in our database have a time field that provides ordering and allows a database worker to determine which sub-table a query should affect, as long at the query itself has a time component. If there is no time-component, then each sub-table has to be checked individually.

| Database Table Name | Description | Is Hyper-table |
|---|---|---|
| Application | Application ID to target URL | |
| Authenticated User | User ID to de-identified user name | |
| IP Type | Campus IP ranges, detailed in table 4.5 | |
| Campus Location | Space ID to GPS | |
| Network Equipment to Location | Network Equipment ID to Space IDs | |
| GeoIPNetwork | Public IP Ranges to GPS | |
| User to Application | Single Sign On login event | X |
| IP to MAC | Internal IP addresses to MAC addresses | X |
| MAC to Port | MAC addresses to switch network ports | X |
| Wireless Auth | User authentications to campus access points | X |

Table 4.3: Listing of database tables and their descriptions, with indicators if table is a TimescaleDB hyper-table.

Five data streams feed into four database hyper-tables. IPv4 to MAC and IPv6 to MAC data from the NetRecon service feed into the IP to MAC table. NetRecon MAC to Port data

feeds into the MAC to Port table in a similar fashion. Both tables have a time stamp and source identifier, and respectively contain IP and MAC or IP and Port fields. Containing a time stamp, user ID, and network equipment ID, the wireless authentication table maintains event status of user accesses to the Virginia Tech campus WLAN. Finally, SSO login events, the focus of our work, provide a mapping of time, user ID, application ID, and source IP.

With each table designed, we can now discuss the connections between the tables that exist. We will follow the perspective of the spatial augmentation process, and all mappings will be based off of the SSO login table. There are four fields within the table. Application ID maps to a URL. User ID maps to a user name and domain. These are the simple mappings needed for augmentation.

### User Location Augmentation

In spatial augmentation the next step is the geolocation of the target user. A combination of the user ID and time provide a time-series based on the most recent user access in the wireless authentication table. This type of relationship, referred to as an *asof* relationship (the knowledge "as of" now), does not have guarantees compared to a regular public/foreign key linkage in SQL. If a user has never been observed on campus, then a null value will be returned. Non-important properties are also lost, such as uniqueness of search.

We consider the asof relationship acceptable for spatial augmentation for its security and interpretation factors. One, this data source is highly trustworthy. Subverting a campus access point or logging server is out of scope for our attack model. Two, this asof relationship will provide data for the majority of users in times of normal operation, since we expect that campus services will be predominantly accessed on campus. Three, the reliability of data is based off of well known factors, such as holidays or system outages.

To confirm the above conjecture on the majority use of campus service, we measure collected data from our first day in the dataset, November 13th, and find that 84%, or 29,116 out of 34,492, of network services users also accessed the wireless LAN. This trend carries out through for normal periods, but changes during holidays. On November 20th, the Monday of Thanksgiving break, only 18% percent, or 3,108 out of 16,830, of network users were also observed on the campus wireless. This rough measure gives credence to the idea that in normal conditions, this information is accurate and useful.

From the mapped wireless authentications we use the network equipment ID to map to a GPS coordinates using the network equipment to location and campus location tables. In SQL, this is an efficient operation using joins. One more value is needed – accuracy. Since each access point serves an area with wireless coverage, and all that is known is the access point's location, then the accuracy level is equivalent to the area of wireless coverage. We use an estimate of indoor wireless coverage at 35 meters and an outdoor coverage of 120 meters. This estimate was made through analysis of an Aruba access point, model AP-255, which at the time the dataset was collected was the predominate type used on the Virginia Tech campus. With exception were a set of four outside access points used by the campus, with an observed maximum range of 120 meters. This fact is not accounted for in the database tables however, we choose to make that check part of the augmentation functions.

**Source IP Geolocation**

Two cases exist for finding the geolocation of a login source IP. Internal campus IPs are mapped using NetRecon information to obtain accurate mappings. External campus IPs are mapped using MaxMind geolocation data. External mappings require a lookup in the preloaded GeoIP Network table, querying based on which range the source IP is in.

Internal campus mappings are created through a set of asof relationships. First, an asof relationship is found between the login and the IP to MAC table. From there, a linkage is built between the MAC to Port table. Compared to the wireless authentication information, there are much stronger guarantees available. NetRecon polling is consistent and provides steady log data. One of the biggest limitations is that the data can be up to five minutes out of date. Dynamic devices, such as phones or laptops, might move around in between the polling period and have a different physical location than indicated by NetRecon information. Additionally, when a new device connects to the system, or changes its IP if dynamic, then no mapping will exist for up to the polling period of five minutes. After that point, a location can be found. Accuracy is found in the same manner to user location accuracy.

Finally, the final piece of information needed for IP geolocation is determining weather a given IP address is internal to the campus or not. This is done using the IP type table, which holds a mapping of all internal IP address ranges with names. Since PostgreSQL and by extension TimescaleDB have built in methods for querying using IP addresses and IP address ranges, this is a straightforward operation.

### 4.3.3 Insertion and Augmentation Functions

With incoming data described and table layout presented, we focus on the implementation of the methods needed to perform loading and augmentation in the database. Our first design choice is to implement the search functionality as internally stored database functions using PL/pgSQL (Procedural Language/PostgreSQL) [40]. PL/pgSQL is an extension of SQL that introduces control structures and allows more complex queries. Using PL/pgSQL provides performance and design advantages. One, round trip time is reduced, since multi-stage queries can be replaced by on client-side call. Two, PostgreSQL takes advantage of the

preloading of functions to do parsing and query planning ahead of time, reducing cost at query tie. Three, this provides a design separation between the query logic and the query itself, making it a more flexible and readable approach. Finally, building the functions this way allows other applications using the database to access these methods, which assists with testing and provides modularity.

A total of seven functions were built to allow parallel loading of the five data streams. PostgreSQL has a built in IP address type, "ipaddr", that allows the abstraction of IPv4 and IPv6 addresses to a single type. This means that the IPv4 to MAC and IPv6 to MAC data streams can be loaded using the same function, loadiptomac. To allow parallelization, we separate out the shared actions needed by the other data streams. For instance, both the SSO login event and wireless authentication data streams need to search for a user ID to match a given user name. And, if that user name doesn't exist yet, add it to the needed table and generate an ID, therefore, the loaduser function is used for loading data from both streams. This separation of common functionality is needed for parallelization, since in PostgreSQL each function is atomic, and locks down tables until completion. Deadlocks occurred when two table actions were integrated together, forcing this segmentation. With the segmentation, the shared aspects are separated out to be accessed in a common way. A description of each function is in table 4.4.

Two functions are implemented for augmentations in the system, one to locate the source IP and its history and the other to locate the user and their location history, described in table 4.4. They operate in the manner in which the spatial augmentation process was described earlier, but simply providing the concrete query mappings needed to perform the necessary table joins. The source IP history function internally conceals two sub-functions, one for locating the internal IP addresses and one for the external IP addresses. The determination is done with an initial table lookup to the list of known campus IP address ranges. Both of

| Purpose | Function Name | Description |
|---|---|---|
| Load | loadapp | Add URL to Application table |
| | loaduser | Add username to Authenticated User table |
| | loaddevice | Add a device to Network Equipment to Location table |
| | loadusertoapp | Load SSO login event |
| | loadmactowifi | Load a Wireless Authentication event |
| | loadmactowired | Load a NetRecon MAC to Port log |
| | loadiptomac | Load a NetRecon IPv4/6 to MAC log |
| Augment | locateuser | Return locations for a user |
| | locateip | Return locations for an IP |

Table 4.4: Listing of database functions and descriptions

these functions are read only, and already parallel capable in our database system by default, since PostgreSQL supports multiple read/writers for a given database table.

None of these functions operate in a batch method, instead they only load a given record at a time. Therefore, these functions can act with no delay in a streaming context, since there will be no wait to fill a given batch of information. Therefore, a function has to be called once per received record in a data stream.

### 4.3.4 Optimizations and Parallelism

Now that the logical functions of the database have been given, we focus on system performance and scalability. Speed of operations is obtained by our design by its simplicity and streaming approach. The data stream method also provides scalability, since separation between hosts can be achieved by segmenting by the data processing, augmentation, and classification functions. Operations would be connected over the network instead of interprocess communications. But for the streaming approach to be viable overall, it must avoid bottlenecking at a given operation. Therefore, we provides a set of straightforward,

but effective performance boosts of naive implementation.

One of the most effective factors providing performance is our table and function design. We build them to function in parallel, so loading and augmentation can occur in tandem and across multiple cores. Indices are built to be optimal for the insertion and augmentation functions. This improves speeds overall since less searching and processing is needed.

The next biggest impact on functionality is our use of the TimescaleDB PostgreSQL extension, that allows efficient queries and loads for time-series data. In our evaluation, we specifically compare the impact of using TimescaleDB versus PostgreSQL without extensions.

Next are a set of other, simple, improvements that we implemented to speed up our performance. The rationale behind doing so was to speed up the development and testing cycle for our work, since the smallest usable unit of data was a day's worth. Our first step was to move the storage disk of the database from a network storage server to a local partition to our test machine. Second, we used the PGTune [1] tool to adjust PostgreSQL settings to best use the amount of RAM and CPUs available in our system. Third, a dedicated worker pool with held database connections was used for loading and augmentation tasks, removing the need to build a new connection per task. These improvements were simple to implement and each provided noticeable speed increases during testing.

In summary of our design, we build a set of tables and functions to support the spatial augmentation process in an streamable and parallel manner. We use a mature relational database in the form of PostgreSQL, providing a free and community supported solution. The next section details the means used to create labeled attacks for training a malicious login classifier.

[1] https://pgtune.leopard.in.ua/

## 4.4 Labeled Attack Creation

Any supervised learning approach requires labeled data for training. For secuirty in general, labels are difficult to aquire, due to massive data imbalance and manual analysis time cost. No set of labled malicious login events suitable for our system exist and especially not in the Virginia Tech camus dataset. Therefore, we synthesize malicious logins events and assume all collected traffic is benign.

Synthesizing attack data is problematic, since past areas have had hidden indicators and bias built in, undoing relavancy of the work. In general, we use the set of known errors from past works to limit bais in our generation approach. Our design is built off of information from two in depth studies in credentail theft and unauthorized logins. [1] provides research with 1.9 billion stolen username and passwords to provide an broad statistics for remote attackers. With [6], more detailed data was collected for over 300 malicious login events. These two works are core to our the threat model description, Section 2.2.

The following rationale was used for assuming that data collected from the Virginia Tech network was benign. Beginning simply, no malicious login events were reported at the Virginia Tech campus in the time period of data that was collected and de-identified. It has been months since the collection occurred, providing time for attack evidence to show itself after the fact. Next, one of the strengths inherent in machine learning based approaches is its inherent generality and are robust to a small percentage of mislabeled data points [27]. If a true attack is mislabeled as benign within the training dataset used, it will therefore not affect model performance greatly.

Restating the threat model developed in Section 2.2, a attacker is an entitiy carrying out a malicious login through credential theft. With the information from both [1] and [6], we determine that an attacker operates in a physically remote location compared to the target

user. At the most basic, adversaries directly access the target account from their remote location. More advanced attackers seek to disguise their location by changing the source of the login event. Even advanced attackers do not have highly accurate location information for their target. Instead, they suffice by choosing an approximately close source to their known target, by subverting on-campus devices, either static or dynamic.

There are four features that must be generated for each synthesized attack: username, timestamp, application, source IP. We work through the generation of each value separately, since there is no indication that these are directly related, i.e. time of attack is independent. From there, we merge these values together in log format, in the same way actual authentication logs are formed, and feed those generated logs into our streaming augmentation process.

One detail is that, while these attacks are generated with random numbers, a random seed is always set at the beginning, allowing for reproducible values. In our evaluation, we report seed values used for each evaluation segment.

### 4.4.1 Username Selection

First a valid username must be selected. We found a set of different options for selection.

- Randomly pick one from the database table "authenticated_users".
- Randomly pick one from a static list of users from all possible days
- Selectively pick a user from the database table who has certain properties such as -
  - On campus for the selected day
  - Had more than one wireless interaction on campus

Our goal in this synthesis is to introduce as little bias as possible in our selection methods. Therefore, we choose to ignore the third option of selectively choosing based on properties. Instead, randomly chosen users will receive additional labels based on what properties they have. Next, we could pick from the loaded table covering a week, or from a static list covering all possible days. We choose the static list approach for two reasons. First, reproducibility is much easier, since random choice will be from the static list. Second, choosing from users that have already been seen would introduce another form of bias.

In conclusion, a static list of usernames was pulled from all the SSO authentication data files, sorted, and condensed down to unique values.In the end though, this came out to 65,892 unique users, though that does include the empty user value. From here, we randomly pick with replacement. Further properties about the user are inferred from this point.

### 4.4.2 Timestamp Generation

Timestamp selection presents an interesting problem. It is one of the most important factors for deciding power of the defense. A midnight timestamp will have different user location than an 11 a.m. timestamp. But, picking times that fit normal human movement patterns also seems to be a losing proposition. We don't know if an attacker would fit that pattern. Based on the work of

For the moment, we choose random selection. In the future, we will improve that approach if a better solution presents itself.

### 4.4.3 Application Selection

This is a feature that could produce a large amounts of bias potentially. Random selection has issues, some applications are very user specific and a model would over train. Picking a single simple value, such as the canvas or banner web login, would also not work for similar reasons. Instead, we pick the top 25 applications, with percentage values (weights), make that list, and pick randomly with replacement. Bias from training on unique applications is preventing, but at the same time, we prevent bias from training on only the top application. We randomly select from this list, with the counts acting as weights.

#### Source IP Selection

This is clearly the option that will produce the most influence, and consequently produce the most potential bias. We will break down the generation analysis by attack case, with each case being randomly selected, then the following procedure being carried out.

#### Remote

For this approach, we use the valuable data collected by [1]. One part of their work provided the top 10 locations of attackers using phishing or keylogging methods. With this, we can randomly select if the attacker is using a phishing or keylogging approach. One note, the study found that a phishing attack is 17 times more likely than a keylogging attack. We randomly select the attack method based on that ratio. Then we randomly select a country based on the given lists. With a country chosen, a Geo2IP table is used to look up a random IP range in that country, and then a random IP is chosen from that range. This reasonable keeps the bias down, while still mimicking the attacker methods.

### Static

A database query provided a list of 1000 static MAC addresses. That data was mapped to IP addresses. IPv6 and IPv4 addresses were both collected, producing a total of 1768 IP addresses. From here, we randomly pick an IP address and use that for the static attacker. The only bias here is that a mapping is guaranteed to happen, so all these IPs will be locatable. Which is quite acceptable, since the only reason some addresses are not locatable is because of issues in the NetRecon service. So we bias towards the assumption that all the requisite services are operational.

### Dynamic

For dynamic attackers, the assumption is that they are using or remotely controlling a random on campus wireless device. Therefore, we use a table of the VT wireless address IP ranges to generate random values. Table 4.5 is a simplified version of what can be found in the database repository.

| Name | CIDR |
|---|---|
| Residence hall NAT (Wired and wireless) | 45.3.96.0/19 |
| Academic NAT (Wireless and privately addressed wired) | 45.3.64.0/19 |
| General Wireless (On-campus use only) | 172.30.0.0/16 |
| General Wireless (On-campus use only) | 172.29.0.0/16 |
| Residential wireless (On-campus use only) | 172.31.0.0/17 |
| Campus wireless networks IPv6 | 2607:b400:20::/44 |

Table 4.5: Internal IP Ranges and Names for dynamic addresses

## 4.5 Model Training and Classification

For classification of malicious login events using the spatially augmented data, we use weighted random forests trained with a combination of benign data and synthetic attacks. The rationale for this model choice are its features of model interpretability and capability of handling unbalanced data. For the implementation of this model, we use the Scikit-learn RandomForestClassifier [43]. Scikit-learn is another open source project for machine learning and provides an intuitive programming interface, making it fit within our design goals.

Model features were already detailed in our previous section on spatial augmentation. One important aspect is we choose to do max/min scaling for feature values. This is chosen because some form of normalization is needed for model performance and we already know the maximum and minimum possible values for each feature from table 4.1. We normalize all values to be between 0 and 1. A brief note is that we did consider numerical instability before this normalization process, particularly for the latitude/longitude values. We find that a float 32 value has roughly seven decimal places of precision, and a GPS value only needs five decimal places to be accurate to the meter. Therefore, our normalization process is also de-normalizable, which benefits when interpreting alert results.

Weighted random forests do not have many hyper-parameters. Of main importance are the number of estimators, followed by class weights, maximum features for a split, and minimum features needed for a split [10]. Instead of relying upon our limited experience with parameter selection, we leverage the speed and parallelization of random forests to perform a grid search of possible hyper-parameter combinations to select an optimal set of parameters and measure the effect parameter variation has on model performance. Our results using this method are detailed in our evaluation. We note that the time-series nested cross validation method was used to provide a robust score immune to data leakage.

### 4.5.1 Analyst Score

A meaningful metric of success is needed when training machine learning algorithms. We translate the goal of reducing analyst cost by formulating a custom training metric that captures this objective. Putting the idea in words, the goal is to limit false alerts, then maximise the recall rate for positive occurrences. Another consideration is that we want an absolute limiting factor upon the number of false alerts, versus a relative one. To do this, we formulate a metric termed the analyst score. The analyst score uses a two external settings. $R$ signifies the amount of false positives for a given temporal period length $P$. For our system, we choose a rate of 2 alerts per hour.

The score is formulated from the combination of the normalized exponential distribution of false positives times the recall rate. Let $l \in L$ be the set of login events within a set period of time $T_E$, and the number of periods within as $c_p$. Let $f(x)$ be a classification function with an associated set of false positive $FP$, true positive $TP$, false negative $FN$, and true negative $TN$ values. Therefore our analyst score is as follows.

$$AS = exp(\frac{-FP}{c_p * R}) * \frac{TP}{TP + FN} \tag{4.1}$$

The first part of this equation is the normalized exponential distribution ($\lambda e^{-\lambda x}$) where $\lambda$ is $\frac{1}{c_p * R}$. Based on the properties of the distribution, the average value is $c_p * R$, which translates to the desired number of alerts per period. In total, this first half provides a measure of the expected alert rate versus the desired alert rate for a given model. In addition, the exponential distribution is skewed left, so the value of a rate less than expected produces a much higher value than one that is more than expected. This provides a measure of elasticity to what otherwise might be a hard threshold value for training.

The right hand of the score is a measure of the classifier recall, or how many malicious events were detected compared to the total number of malicious events. This allows training methods to optimize on the number of detected attacks after limiting the number of erroneous alerts. Since both of these values lie between 0 and 1, the resulting score is also between 0 and 1.

We note that interpretation of this score varies from normal, since a score of 0.5 implies either that all positive events were identified within the expected alerting rate, or no false alerts were thrown and 50% of positive events were identified. Either of these properties are desirable from the analyst perspective.

To review, event classification presents a design challenge of generality. Augmentation of logins produces a set of continuous contextual features. For interpretability, we restrict our set of potential models to any that are decision tree based. Random forests fits design goals of simplicity and speed, making it the chosen model for our implementation. But, we emphasize here that any model capable of meeting design goals would fit. An example would be boosted trees [28]. Even though the design is model agnostic, care should be and is taken during implementation to avoid over fitting and use high precision as a metric of training.

## 4.6 Chapter Summary

In this chapter, we first provided a set of goals for systems that seek to implement a login classification ability. We presented the subject of spatial augmentation, process of using dynamic network information and static location data to provide user and IP trajectories for login events. From this, we described a design for a streaming database for implementing this functionality in a timely manner. After augmentation, we receive records suitable for algorithmic classification. While our approach is model and training agnostic in general,

for our proof of concept we choose to use weighted random forests trained using time series nested cross validation on a set of benign events and synthetic attacks. To build a model that reflects our system goals, we provide a custom scoring metric, the analyst score, that uses an absolute metric of false alerts in combination with model recall to provide a measure that reflects needs of security personnel.

# Chapter 5

# Evaluation

Assessing if our detector implementation meets and addresses the overall problem in general, and design goals in specific is a multi-stage process. Following the path of data flow, we first assess the dataset used for training and further evaluation, checking for preexisting anomalies and consistency of data. The purpose of measuring the dataset is to acknowledge and prevent potential issues in the further model evaluation. After establishing the data validity, we measure the load time requirements of our TimescaleDB database and the associated loading, augmentation, and classification functions that operate through it. Since these processes operate in parallel, we first test the individual speed of each component, then measure the overall speed of these operations. An important point is that we assess the short and long term trends of the streaming processes. Finally, we evaluate our random forests model for classification accuracy and generalisability, using our created attacks. We perform a grid-search across the model hyper-parameters to determine both the optimal settings for the model and how the model performs for suboptimal settings. Finally, we compare this process to the commercial tool GeoLogonalyzer, which also aims to alert on anomalous login events using location based attributes.

## 5.1 Static Dataset Assessment

Rationale for our evaluation approach is based on the aim to produce a detector that is both performant and useable in a campus environment. Dataset assessment is therefore our first testing step. Other works [44], have been invalidated or weakened due to failure to thoroughly examine their dataset being used for training and testing. One such dataset is the KDD99 dataset, which has age and internal consistency limitations.

Taking a similar approach to [44] we first assess the consistency of the data flows, then determine any inconsistencies in the data itself. By taking this route, we find and handle these issues before proceeding with model training or database evaluation. Handling an identified issue might just require acknowledgement. For instance, a logging node or system failure might inhibit the flow of data to the detector, reducing classification ability for a period of time. But, as long as our processes can handle empty values, the detector itself will not be severely affected. Furthermore, because we had already acknowledged this failure in the dataset, the change in model accuracy and database speed will be correctly interpreted as dataset failure, as opposed to instability in our system.

The following behaviors are of interest to our dataset assessment, since they are indicative of either network failure, a campus-wide event (e.x. Thanksgiving), or a change in network setup.

- Times of data stream loss
- Order of magnitude variations in data stream flow
- Non-periodic variations in record count

We proceed through a manual analysis of the daily and hourly counts of the collected data, shown in Figure 5.1. In addition, Table 5.1 provides a listing of special events that occurred

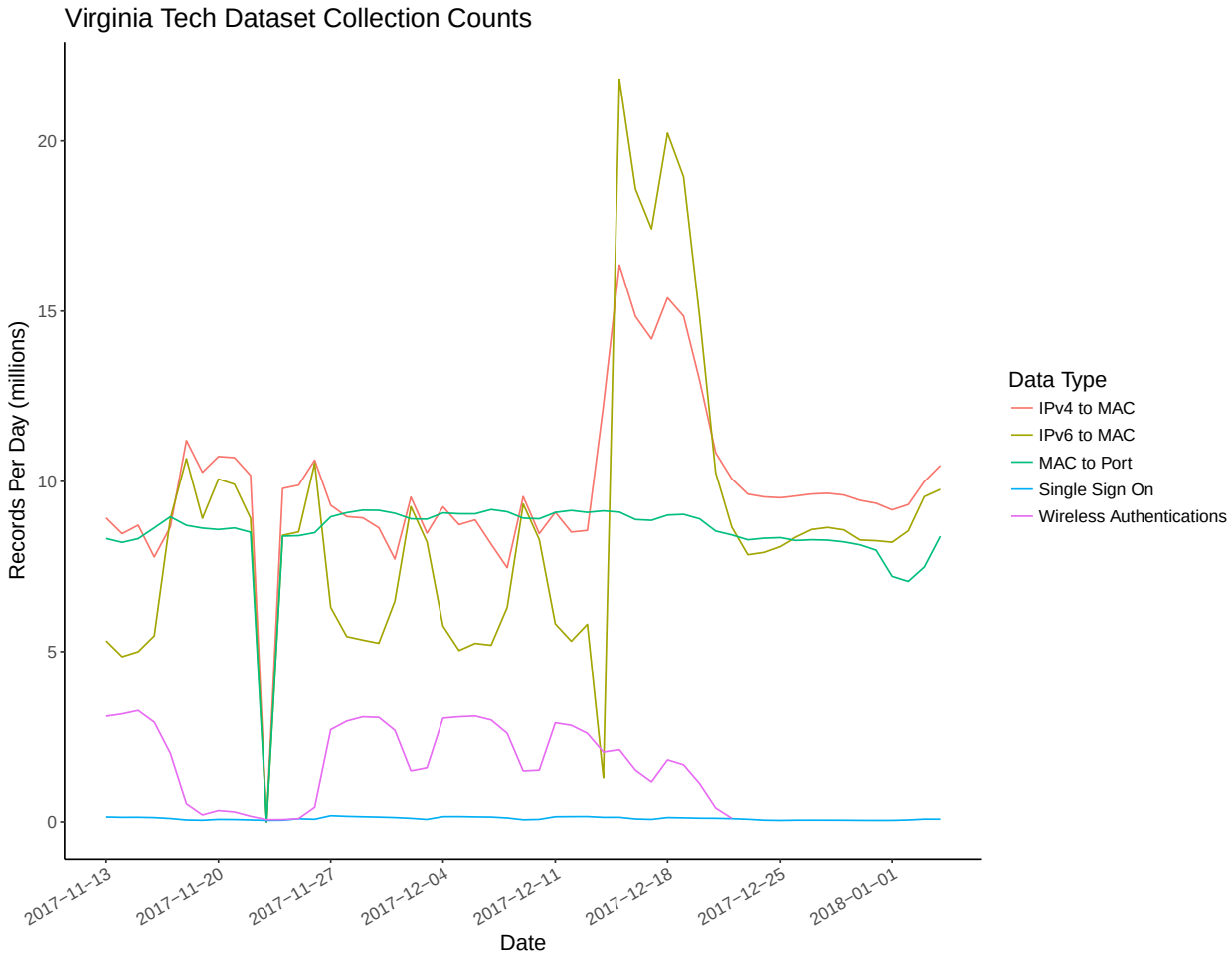


Figure 5.1: Chart of the amount of dynamic records collected each day from the Virginia Tech network. Each time mark is a Monday, representing the start of a week.

on the campus during the collection time period. Correlating these two sources of information, we can see that periodic events such as weekends, special events, and winter break have an effect upon the amount of data recorded.

Several anomalous aspects are apparent. The most important and interesting one is that while the wireless authentications correlate well with expected user activity, all the NetRecon data has an inverse correlation with human movements. Lows in the amount of wireless authentications are weekends. The first large low is the campus week-long Thanksgiving break. This is sensible, since less people on campus would be much less devices authenticating

| Week Number | Week Start | Events |
|---|---|---|
| 1 | 2017-11-13 | None |
| 2 | 2017-11-20 | Week-long Thanksgiving Break |
| 3 | 2017-11-27 | None |
| 4 | 2017-12-04 | None |
| 5 | 2017-12-11 | Last Day of Classes, Start of Exams |
| 6 | 2017-12-18 | End of Exams, Graduation |
| 7 | 2017-12-25 | Christmas, New Year's Eve |
| 8 | 2018-01-01 | New Year's Day |

Table 5.1: Important events that occurred at Virginia Tech in the period of November 13th to January 4th.

across the network. In contrast, NetRecon sources, such as IP to MAC and MAC to Port, spike as user activity decreases. After investigation, we find two factors causing these spikes to occur. First, NetRecon is a polling operation with the goal of polling the network state every 5-10 minutes. This service operates in a self throttling manner, and will only poll less in times of high user activity, as low as every 10 minutes. Another factor is that in times of lower activity, switches keep their internal IP mappings for longer. So after a connected devices leaves the area, the mapping might be logged for another period. Both of these factors account for the upswing in logged events during Thanksgiving and weekends. But, other factors influence the large spike that occurs on week 6 and after, when end of exams and graduation happens. More switches were added into the NetRecon system and polling rules were changed, so more information was collected, less mappings were dropped, and polling was consistent.

Less interesting, but equally anomalous is the loss of the wireless authentication data after December 22nd. This occurred due to a change in data format and our collectors did not collect any data past this point. Therefore, we address this issue by constraining our analysis to the period when wireless authentication data was available. This limits our useful data to only 39 days of valid information. Another loss occurs on Thanksgiving day, November

23rd, where the NetRecon service had a system failure. Logins and wireless authentications were still collected however. We expect poor classification ability on that day, since there will be a loss of information.

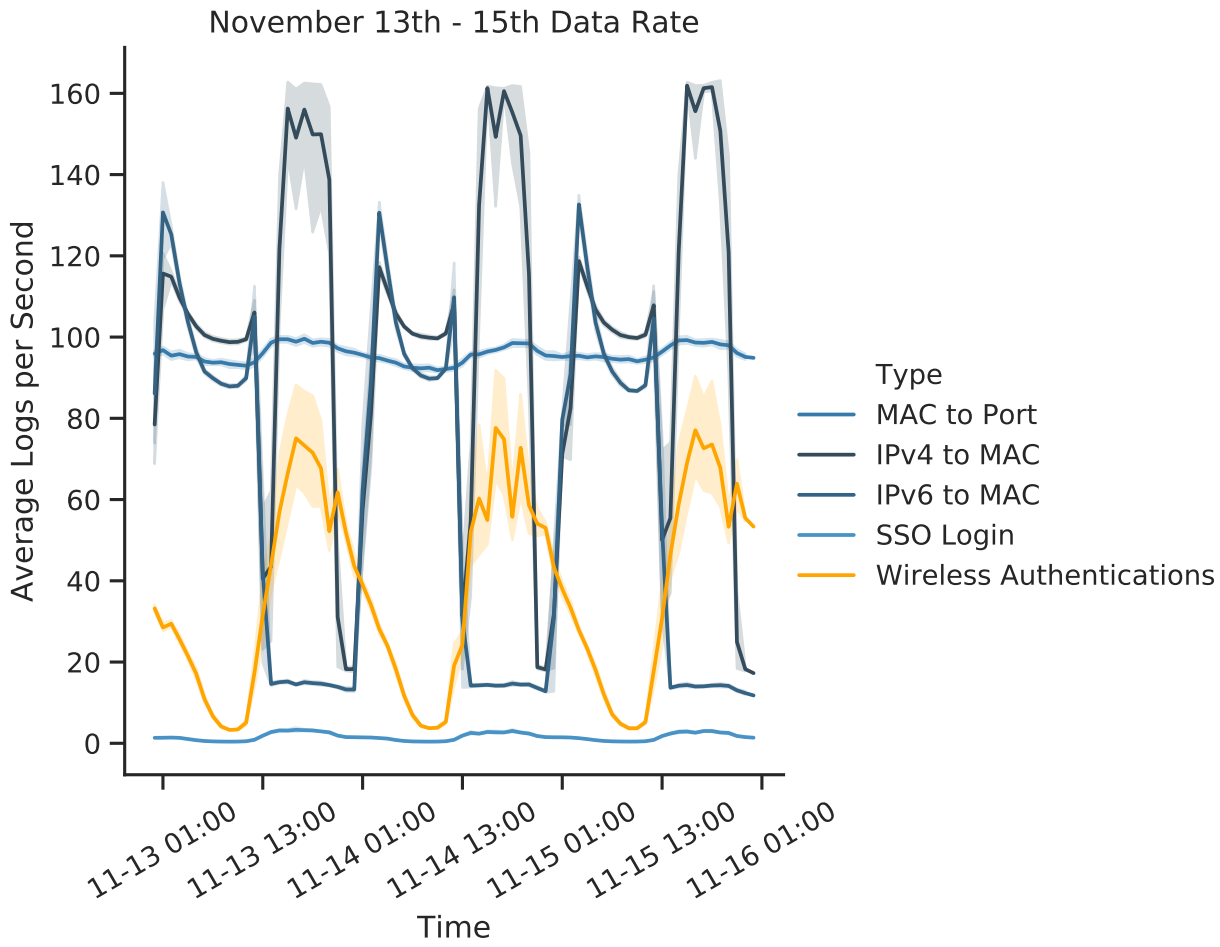


Figure 5.2: Graph of the amount of logs per data type in a five minute period on the day of November 13, 2017.

After considering the data view on the daily level, we focus more on how the data acts minute to minute. Figure 5.2 is a graph of the rate of log events received for a given data stream each hour on in the given time frame. There are several aspects to note about this data. First, shortly around 12:30 P.M., the amount of IPv6 to MAC data collected sharply drops off and the IPv4 to MAC data becomes inconsistent. This drop off is the previously

mentioned rate limiting that occurs during periods of high user activity. This is corroborated by the increase in the number of wireless authentications, which are events of device arrival or movement. The spikes in the authentication rates correlate to times of class change in the university, which are known periods of high user movement. While not evident in the graph, the rate of login events also spikes in a similar fashion to the wireless authentications. In addition, we have that the MAC to Port mappings are highly consistent, implying that the majority of IP addresses assigned are dynamic and from wireless access points, compared to use of wall outlets for wired connections. Even though they are consistent, they still vary with user activity, meaning that a portion of users attach devices to the network daily.

In summary, the following issues and differences can be expected across this dataset:

- Four weeks of normal operational data were collected, four others were affected by special events.

- NetRecon data was lost for the entire day of Thanksgiving, November 23rd.

- NetRecon polling events are inversely proportial to the amount of user activty, this issue was fixed near the end of data collection.

- Reduction of NetRecon data rates only occurs during periods of high user activity, making the rest of the day reliable.

- Wireless Authentication information was not received after December 22, making that the end of our effective testing period.

| Language | Packages/Modules | Purpose |
|---|---|---|
| Python 3.6.4 | Psycopg2 2.7.5 | PostgreSQL Adapter |
| | Scikit-Learn 0.19.2 | Machine Learning |
| | Pandas 0.23.4 | Data Analysis |
| | Numpy 1.15.0 | Matrix Operations |
| | SciPy 1.1.0 | Scientific Computing |
| | SeaBorn 0.9.0 | Statistical Data Visualization |
| PostgreSQL 10.5 | PL/pgSQL 1.0 | Internal Procedural Language |
| | TimeScaleDB 0.10.1 | Time-series Data Extensions |

Table 5.2: Software Environment Packages

## 5.2 Test Environment

A consistent test environment was used for all of the following experiments. Hardware and OS details are in table 5.3. Software language and package version information is in table 5.2.

While 32 threads were available for use in our test machine, in general, only 16 cores were used when running, since initial tests only utilized 16 cores. We make clear for each evaluation step the number of cores used. Additionally, some tests were single core only, and that fact is made clear.

These software package versions were the most up to date for our project at install time and were deliberately not updated for the entire evaluation.

## 5.3 Streaming Data Processing

Database and associated data processing task performance are the next logical sections to evaluate. For each streaming function – data loading, spatial augmentation, and model classification – we evaluate it's speed and reliability. We test the functionality of our table

| Specifications | | Testing Environment (VM on Dell PowerEdge R620) |
|---|---|---|
| CPU | Manufacturer | Intel |
| | Model | Xeon E5-2670 |
| | Frequency | 2.6 GHz |
| | Sockets | 2 |
| | Cores | 32 |
| | Threads | 32 |
| | L3 Cache | 20 MB shared |
| Memory | Type | DDR3 |
| | Capacity | 32 GB |
| | Frequency | 1600 MHz |
| Storage | Manufacturer | Dell |
| | Model | H710p |
| | Type | SAS RAID 5 |
| | Capacity | 250 GB |
| Operating System | Version | XenServer 7.2 (CentOS 7 VM) |
| | File system | EXT4 |

Table 5.3: Test Machine Specifications

and function design. Before evaluating the final streaming model, we first evaluate the ability of the TimescaleDB module for long term load performance.

### 5.3.1 Comparison of TimescaleDB to PostgreSQL

To begin, we evaluating the algorithmic difference in load and augment times between a basic PostgreSQL database and one with the TimescaleDB extension. The purpose of this evaluation step was to evaluate the ability of the TimescaleDB extension. We use the first week of normal data from our dataset for testing. Previously mentioned, this is the week before Thanksgiving, so there is a drop off in total number of records per day as the week goes on. Therefore, we use the metric records per second (RPS) to signify the calculated amount of log records processed or extracted.

In this case, we measure the loading and augmentation processes only, since classification speed is unaffected by the speed of the database itself. When this initial testing occurred, our design and code implementation had not yet been implemented. Therefore, all loading functions are run on a single thread, and no worker pools with held connections or database tuning features were applied. At this time however, there was a desire to speed up the loading process to usable speeds and streaming evaluation was low priority, so a connection batching process was being used. All individual operations were per row and streaming capable, but each connection was loaded with a batch size of rows and associated functions, before being committed to the database. For these tests, the batch size was set to 100,000 rows per connection. Augmentation was still streaming based and applicable. These evaluation steps also did not used any form of step-wise measurements, meaning that the only metrics were time to load a set of data and the amount of data loaded.

Table 5.4 shows the averaged rows per second speed needed to load all the static data, a

| Load RPS | PostgreSQL | TimescaleDB |
|---|---|---|
| Static | 5047.6 | 2016.0 |
| Load Day | 843.7 | 3185.8 |
| Load Week | 277.5 | 3745.2 |
| Augment Day | 2.40 | 17.78 |
| Augment Week | 2.01 | 17.65 |

Table 5.4: Average load rate in records per second for static data, one day of dynamic data, and a week of dynamic data, for a PostgreSQL database with and without the TimescaleDB extension.

day of dynamic data, and a week of dynamic data for both database systems. For the static data, PostgreSQL does performs much better, but in contrast performs significantly worse for the dynamic time-series data. In total, loading the PostgreSQL database took 3 days and 15 hours, compared to the Timescale version which took less than 15 hours to load. One note about the loading times for the PostgreSQL database, they drop logarithmically, from roughly 800 RPS on the first day, to 400 RPS on the second, and 200 RPS on the fifth day.

After data loading must come login augmentation. For this process, we use 16 cores with no other operations occurring during augmentation. One note is that we limit our core usage to 16 cores for our Python augmentation program. But, the database also uses cores when operating, which might vary run times. However, for PostgreSQL and consequently TimescaleDB, each connection process has an associated database process. From this aspect and system observation only 16 cores were ever in use at the same time when running these tests. Again referencing table 5.4, we see that load speed for the TimescaleDB is significantly higher. And operation speed for both databases does not change appreciably over time.

Interpreting these results, we confirm the reasoning behind choosing the TimescaleDB extension. As PostgreSQL table sizes grow, insertion cost increases logarithmically, which is normally the optimal configuration for a database. But, since TimescaleDB has awareness of the properties of time-series data, table index sizes are reset every day, preventing this

decrease in scaling. This explains why the load times don't decrease over time. The speed increase is provided by TimescaleDB optimizing for the common case of new records being directly after a previous record. This allows table indices and storage optimizations that would have previously been impossible in the general case that PostgreSQL seeks to answer. The augmentation speed difference is also significant, but doesn't noticeably decrease over time.

Now that the use of the TimescaleDB extension has been evaluated, we turn our focus to assessing the streaming implementation of loading, augmenting, and classifying our data.

### 5.3.2 Data Loading

We now present the results of loading three days of data into the database in a pure streaming fashion. Specifically, we insert logs from November 13 - 15th into the database. To collect accurate timing information that fully captures the periodic variations of data flow, we segment the data by a temporal period. For this analysis set, a period of five minutes was chosen. Therefore, a total of 288 readings per day of loaded data were collected, with each reading being a consecutive five minutes of data. This approach provides the ability to grasp performance on times of peak data and times of minimum data in the day. Based on the previous dataset evaluation, we know that peak data flow is dependant upon the level of user activity on the campus.

In this analysis stage, each five minute task was given to a single processor core. The final result was one processor core per data type for either reading in five minutes of data or loading five minutes of data then an additional core for augmentation and a final core for classification. Therefore, only 12 cores were ever in use for a given five minute period of data. However, for testing purposes, we allow more temporal groups of data to be added

| Data Rate | IPv4 To MAC | IPv6 To MAC | Wireless | MAC to Port | SSO Login |
|---|---|---|---|---|---|
| Mean | 100.6 | 58.5 | 36.7 | 95.8 | 1.6 |
| Std. Dev. | 45.7 | 43.9 | 27.4 | 2.5 | 0.9 |
| Min | 16.9 | 11.6 | 2.7 | 90.0 | 0.3 |
| Max | 166.7 | 145.6 | 132.4 | 102.6 | 5.8 |

Table 5.5: Statistics on amount of received logs per second

| Load Rate | IPv4 To MAC | IPv6 To MAC | Wireless | MAC to Port | SSO Login |
|---|---|---|---|---|---|
| Mean | 1165.2 | 1164.8 | 584.9 | 733.6 | 580.6 |
| Std. Dev. | 78.0 | 84.3 | 38.8 | 45.9 | 51.8 |
| Min | 1025.9 | 879.5 | 494.7 | 645.8 | 394.3 |
| Max | 1761.3 | 1748.6 | 748.4 | 931.7 | 852.1 |

Table 5.6: Statistics on amount of logs loaded per second for each type of received data.

to the internal task queue, meaning that the other available cores will be used for the next five minutes of data while the original five minutes of data are being loaded. Since timing is done per core, metrics are not affected. This does provide the knowledge that our measured times are for a database at peak usage, adding meaning to our test results.

For loading data, we collect and report on the load speeds for all five data stream types, since each stream has it's own daily variations and load characteristics. To begin, table 5.5 shows the log arrival rate for the test period of time in the metric of number of received logs per second. These numbers provide the reference information needed to judge the needed speeds for our database loading abilities. The goal of our database is to meet or exceed these values, allowing all received information to be stored with minimum delay.

Table 5.6 is a direct contrast to table 5.5. For every data type, the minimum load rate for the data type exceeds the maximum receive rate. This shows that our system is able to keep up with the amount of data received in a very stable way and that the period peaks of data supplied will not cause the system to fail in an unexpected manner.

| Load Time | IPv4 To MAC | IPv6 To MAC | Wireless | MAC to Port | SSO Login |
|---|---|---|---|---|---|
| Mean | 26.1 | 14.8 | 19.1 | 39.3 | 0.8 |
| Std. Dev. | 12.2 | 11.0 | 14.7 | 2.6 | 0.5 |
| Min | 2.9 | 2.1 | 1.3 | 30.9 | 0.1 |
| Max | 45.6 | 39.1 | 69.4 | 46.1 | 3.0 |

Table 5.7: Statistics on data stream load times in seconds for every five minute period of data. .

Table 5.7 provides an absolute measure of the time it takes for each data stream to load a five minute period of data, values being in seconds. This metric is provided first, since the relative measure is only important if the absolute measure is reasonable. Since each stream is loaded in parallel, the maximum load time across the stream types is the limiting factor. Unsurprisingly, the stream in question is the wireless authentication stream, which is one that is most sensitive to user activity. The maximum amount of user movement occurs at 6:50 on Tuesday, November 15th, when the university class period ends and many students are studying on the campus. The number of actual received events is 39,279 logs in five minutes, compared to the overall average of only 11,000 events. Still, in the worse case, our system manages to load five minutes of received data in less than 70 seconds.

Finally, 5.3 is a plot of the processed rows per second of our data base for each five minute period in the tested time. The first, and most important feature is that every day, at midnight, the load speed sharply spikes up. This is due to the TimescaleDB database creating a new table for the given data type. We can actually see that the load speeds have a logarithmic decrease in speed over time, but the daily table refreshes prevent this speed reduction from becoming a problem. Another aspect is the three levels of load speeds. This is based on the number of database operations needed to add a record to our system. For IP to MAC data, only one database operation is needed. MAC to Port requires two sequential database operations, while loading both SSO login and wireless data streams need three

database operations.

It can be seen that the level of user activity did not affect the rate at which records were loaded. No periodical rise and fall of information occurs, just a continual flattening curve and a refresh spike.

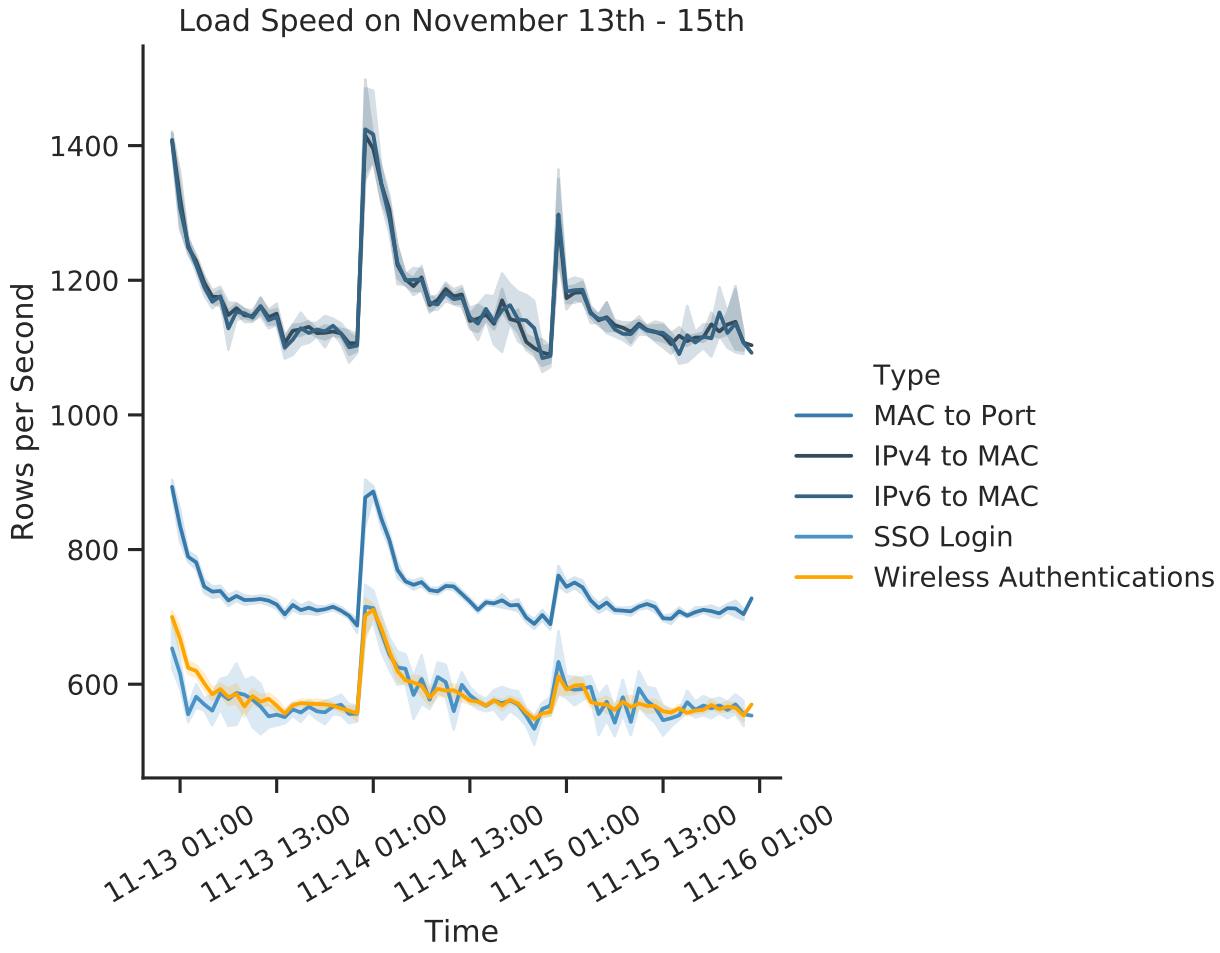


Figure 5.3: Processing speed of the database system, how many log events are being loaded per second. SSO Login times additionally include the cost of augmenting and classifying the login event.

### 5.3.3 Login Augmentation

After establishing the absolute and relative loading rates, we move to the next step in our system pipeline – login augmentation. This step is by definition the slowest, since two complex database queries occur along with streaming based min/max normalization.

Interpreting figure 5.4, we have several conclusions to draw. First, since this query looks back in time over a day, we can see that there is an expected initial speed that quickly drops to a more reasonable rate. After a full day is loaded, the values level off at an average of 4 logins augmented per second by a single worker process. This speed is enough for the average case, but would be a limitation during peak times. Of course, this process is parallel capable, and another worker thread can be started as need be.

## 5.4 Model Performance

Model classification and associated alert generation abilities are next to be assessed. To do this, we train a set of weighted random forests with stepped variations in hyper-parameters in a process called grid search, where each model's ability is assessed with time-series nested cross validation. For the cross validation phase, a cut off metric of is used for the number of generated alerts per day, while the grid search optimizes for the highest precision withing that space.

For our purposes, we used a week of augmented login events, from November 13th - 19th. We generated 7000 malcious events for training purposes, randomly distributed and built according to our previously discussed methods. Table 5.8 shows the parameters adjusted to perform our model setting grid search. For testing purposes, the generated attacks were downsampled each test and validation period to reflect realistic events, with a set amount

| Parameter | Values |
|---|---|
| Estimator Count | 8, 16, 32, 64 |
| Max Features per Node | 2, 4, 8, 16, 32 |
| Minimum Samples per Split | 2, 4, 8, 16, 32 |
| Max Tree Depth | 8, 16, 32, 64, No Limit |
| Class weights | 1:1, 1:100, 1:10000, Balanced |

Table 5.8: Grid search parameters for weighted random forests classifier, totalling 2000 different combinations.

| Day | False Positive Count | Recall Rate | F-Beta Score |
|---|---|---|---|
| November 14th | 48 | 78.36% | 88.9% |
| November 15th | 36 | 76.63% | 89.6% |
| November 16th | 30 | 79.46% | 91.1% |
| November 17th | 23 | 79.07% | 92.0% |
| November 18th | 105 | 75.00% | 80.8% |
| November 19th | 24 | 74.36% | 90.4% |

Table 5.9

of attacks each day, including the case where no attacks occured in a given day.

Using the time series nested cross validation approach We used an outer split of seven, making six training cases with the next day being the test day. For our inner parameter tuning and validation, we used a time based split of eight hours per day, making more training cases as time went on. Table 5.9 shows the final test scores for the six data folds.

An expected decrease in performance is on Saturday, November 18th, when user activity starts to differ as users leave for Thanksgiving Break, model performance suffers significantly. Interestingly enough, the performance has improved again by the next day. This shows that it is the widescale change in user location, as in the majority of students going home, that affects ability, not the actualy majority location. After user activity stabalizes on the 19th, where students are home and have stayed home for a day, the classifier ability also improves.

## 5.5 Comparison with GeoLogonalyzer

As a final measure of the system classification ability, we compare performance of the system to the tool GeoLogonalyzer by FireEye [45], a recently developed solution for detecting malicious logins, that operates using user login history and external IP geolocation. GeoLogonalyzer's approach is simpler than our system, only using the broader geolocation provided by MaxMind services [36], and only considering information from the login event stream itself. However, this detection tool is the most advanced solution available at this time, making it the prime candidate for comparative evaluation to our system's approach.

For testing, we updated the tool to Python 3.6 and adjusted the input records to match our data. Additionally, we set the default location for all internal IP addresses to be the Virginia Tech campus. We use same week of data and associated attacks that was utilized in our model training and evaluation section, assuring consistency. No other modifications to the tool were made.

Table 5.10 shows the final scores the tool acheived for our week of test data. Unfortuantly, it nearly threw as many alerts as there were logins, showing poor performance. Analyst scores that are zero were not computable, becoming too low for double precision floats. In total, the ability of this tool compares highly unvaforably with the ability of our system.

## 5.6 Chapter Summary

Evaluation begins with analysis of the dataset in use, identifying issues relevant to malicious login detection, irrespective of classification method. The environment used for testing is then detailed for future reproducibility. Results are first presented on the system used to provide the spatial augmentation capabilities. Testing using the TimescaleDB extension

| Day | False Positive Count | Precision | Recall |
|---|---|---|---|
| November 13th | 72,890 | 0.39% | 28.4% |
| November 14th | 69,920 | 0.41% | 28.7% |
| November 15th | 72,038 | 0.38% | 27.4% |
| November 16th | 64,034 | 0.36% | 23.6% |
| November 17th | 53,434 | 0.33% | 18.2% |
| November 18th | 34,538 | 0.35% | 12.7% |
| November 19th | 31,377 | 0.28% | 8.4% |

Table 5.10: Result information for the GeoLogonalyzer tool over a week of Virginia Tech data. Zero values are indicative of numbers too low for our computational accuracy. NOTE - these values are missing due to an error in running this tool earlier, this process is being rerun now.

shows its long term scalability and a fully optimized database is shown to maintain on average 10x the data ingest speed needed for the Virginia Tech campus. Spatial augmentation processing speed is also shown to maintain parity with data inflow rate, while classification and alert generation costs are negligible. Model performance is measured through an extensive grid search of parameters for remote, static, and dynamic attacks on the first week of augmented data, where a few attacks are used for model validation and final test scoring. Final comparison of measurements is through comparing remote attack alerting capabilities of the GeoLogonalyzer tool with this work's trained model. Classification of remote attacks by our trained models is shown to simply meet stated needs of low alert rate and higher recall. Static and dynamic results do not, though show promise. In comparison, GeoLogonalyzer generates three magnitudes more false positives each day of operations, failing to fulfill the needed detector design goals.

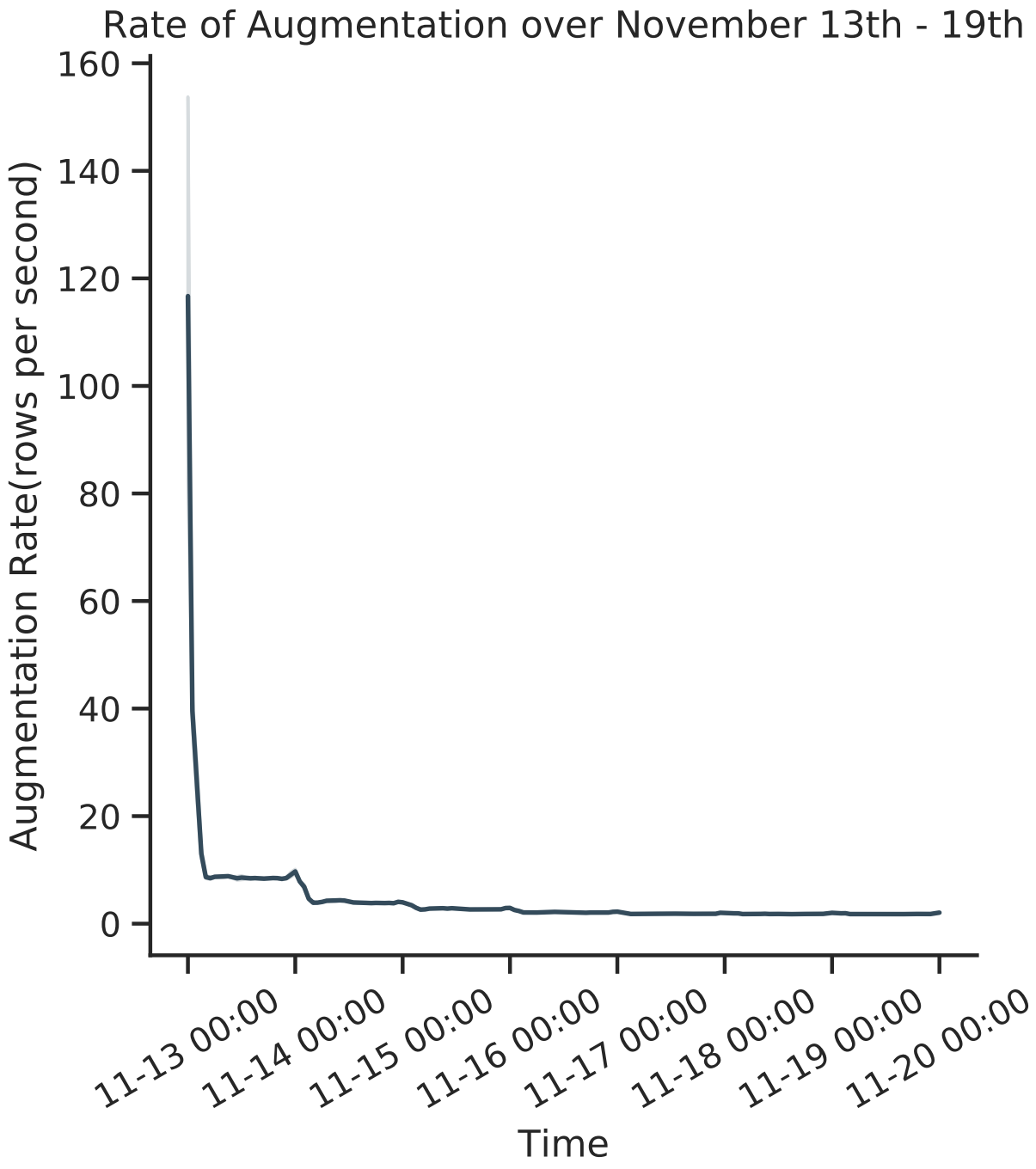


Figure 5.4: Chart of the logins augmented per second over the first week of data.

# Chapter 6

# Conclusion

Credential theft and other methods provide attackers with a variety of means to subvert network defences. Higher educational institutions are particularly vulnerable, with BYOD policies and population variety. Preventing password theft is infeasible for this situation, but detecting a malicious access with the stolen information is possible. If possible, a solution must not simply identify suspicious events, but accommodate analyst needs with system responsiveness, interpretability of alerts, and time cost limitations.

Spatial data, collected though the knowledge of campus network equipment, provides the context needed to classify service login events as malicious or benign, with enough information. We investigate the ability of this data to effectively detect adversarial actions, while keeping within environmental limitations of a campus network.

## 6.1 Contributions

Beginning with the central idea, we have designed and implemented a method of spatial augmentation that provides location information for service login events in a campus network. We then carry out the evaluation of this concept in a variety of means.

First, from a responsiveness standpoint, we program a proof of concept system that performs this data augmentation and the subsequent classification in seconds. A pipeline of data

processing is built and is shown to perform operations at the needed speeds or higher to maintain parity of the data flow of the Virginia Tech campus network. We show that this approach holds for extended period of time, using line data collected and handled under an IRB approved de-identification protocol.

Model training is done using a combination of synthetic attack data and the collected benign data. Using results from multiple studies of this attack vector, we build an event generation method for three classes of malicious logins. These classes are defined by us as remote, static, and dynamic and assume a competent attacker with more resources and time between each attack case. We introduce a new means of measuring model ability with a metric termed the "analyst score", which is a combination of absolute false positive rate over time and the total model recall. This metric fulfills the dual needs of alert rate reduction and high accuracy alerts.

This research provides initial exploration of the concept of malicious login classification with spatial data. While other solutions and techniques exist for other environments, none are applicable within the constraints of a campus network, making this project unique.

## 6.2 Limitations

Based on our evaluation and the numerical findings within, several aspects of our work could be improved. The largest problem is that loss of NetRecon IPv6 to MAC mappings during high periods of user activity reduced the level of useable data dramatically. This had a strong impact on our model dynamic attack classification ability and a possible lesser impact on analyst score for other attack cases. Additionally, in general a higher polling rate would provide better information for our system, with more state changes being captured.

Other limitations include the need to generate malicious events for model training and evaluation purposes. While we leverage the knowledge gained from other studies to build a strong generator, there is still the intrinsic fact that we do not evaluate against a truly motivated and intelligent adversary. From there, we assumed the integrity of the received data, which was reasonable within the scope of our model. But, methods such as MAC or IP address spoofing might fool our system and would be available to a skilled enough attacker that had gained internal network presence.

Another limitation is that our system could not be evaluated on real-time campus data. Due to the IRB data handling protocol, the de-identification methods used were not streaming capable. Additionally, while real time network log collection is implemented at the Virginia Tech campus, the capabilities to provide it for real-time research usage are still in development.

## 6.3 Future Works

Prominent in our future works is considering other models for login event classification combined with additional data sources. Deep learning methods might lend themselves to this problem, providing a valuable boost in the analyst score metric while sacrificing interpretability. Additionally, other data sources are available for this work, such as graph connections of network devices, the target applications and their user groups, VPN and network traffic logs to locate off campus user activity, and wireless association data compared to wireless authentication events. All of these data sources would provide either better location data or contextual information that would lead to better model performance. Additionally, user movement patterns could be inferred through a data embedding, allowing better predictions for new and uncommon users to the campus network.

Further reliability, scalability, and classification evaluations need to be done to guarantee that this approach is as practical as theorized. Further development needs to be done for error handling automatic nightly model training.

The spatial data inferred by this system might have other applications in the field of computer security and beyond. User movement patterns, both individual and group, might lend themselves to sociological pursuits. Additionally, this design would be implementable for an enterprise campus or city wireless network, providing both security and awareness for both environments. Of course, other factors and obstacles would need to be addressed before this could happen.

# Bibliography


[1] K. Thomas, A. Moscicki, D. Margolis, V. Paxson, E. Bursztein, F. Li, A. Zand, J. Barrett, J. Ranieri, L. Invernizzi, Y. Markov, O. Comanescu, and V. Eranti, "Data breaches, phishing, or malware?: Understanding the risks of stolen credentials", ACM Press, 2017, pp. 1421–1434, ISBN: 978-1-4503-4946-8. DOI: 10.1145/3133956.3134067. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3133956.3134067 (visited on 01/29/2018).

[2] R. T. Verizon, "2017 data breach investigations report", Verizon, 2017. [Online]. Available: http://www.verizonenterprise.com/verizon-insights-lab/dbir/2017/.

[3] D. o. Justice, "Nine iranians charged with conducting massive cyber theft campaign on behalf of the islamic revolutionary guard corps", Justice Department, Press Release 18-350, Mar. 23, 2018. [Online]. Available: https://www.justice.gov/opa/pr/nine-iranians-charged-conducting-massive-cyber-theft-campaign-behalf-islamic-revolutionary (visited on 04/29/2018).

[4] 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)*, Oct. 2016, pp. 1–8. DOI: 10.1109/VIZSEC.2016.7739582.

[5] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection", in *2010 IEEE Symposium on Security and Privacy*, May 2010, pp. 305–316. DOI: 10.1109/SP.2010.25.

[6] J. Onaolapo, E. Mariconti, and G. Stringhini, "What happens after you are pwnd: Understanding the use of leaked webmail credentials in the wild", in *Proceedings of the 2016 Internet Measurement Conference*, ser. IMC '16, New York, NY, USA: ACM,

2016, pp. 65–79, ISBN: 978-1-4503-4526-2. DOI: 10.1145/2987443.2987475. [Online]. Available: http://doi.acm.org/10.1145/2987443.2987475 (visited on 06/14/2018).

[7] Grant Ho, Aashish Sharma, Mobin Javed, Vern Paxson, and David Wagner, "Detecting credential spearphishing attacks in enterprise settings", in *Proceedings of the 26th USENIX Security Symposium*, USENIX Association, Ed., OCLC: 255334142, Vancouver, BC, Canada, Aug. 16, 2017, pp. 469–485, ISBN: 978-1-931971-40-9.

[8] B. Morrow, "BYOD security challenges: Control and protect your most sensitive data", *Network Security*, vol. 2012, no. 12, pp. 5–8, Dec. 1, 2012, ISSN: 1353-4858. DOI: 10.1016/S1353-4858(12)70111-3. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1353485812701113 (visited on 05/16/2018).

[9] H. Siadati and N. Memon, "Detecting structurally anomalous logins within enterprise networks", in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '17, New York, NY, USA: ACM, 2017, pp. 1273–1284, ISBN: 978-1-4503-4946-8. DOI: 10.1145/3133956.3134003. [Online]. Available: http://doi.acm.org/10.1145/3133956.3134003.

[10] L. Breiman, "Random forests", *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, Oct. 2001, ISSN: 0885-6125. DOI: 10.1023/A:1010933404324. [Online]. Available: https://doi.org/10.1023/A:1010933404324 (visited on 07/20/2018).

[11] P. A. A. Resende and A. C. Drummond, "A survey of random forest based methods for intrusion detection systems", *ACM Comput. Surv.*, vol. 51, no. 3, 48:1–48:36, May 2018, ISSN: 0360-0300. DOI: 10.1145/3178582. [Online]. Available: http://doi.acm.org/10.1145/3178582 (visited on 08/11/2018).

[12] A. Hiltgen, T. Kramp, and T. Weigold, "Secure internet banking authentication", *IEEE Security Privacy*, vol. 4, no. 2, pp. 21–29, Mar. 2006, ISSN: 1540-7993. DOI: 10.1109/MSP.2006.50.

[13] D. Hardt, "The OAuth 2.0 authorization framework", Internet Requests for Comments, Technical Report 6749, Oct. 2012. [Online]. Available: https://tools.ietf.org/html/rfc6749#section-1.3.1 (visited on 08/15/2018).

[14] S. Willens, A. C. Rubens, C. Rigney, and W. A. Simpson, "Remote authentication dial in user service (RADIUS)", Internet Requests for Comments, 2865, Jun. 2000. [Online]. Available: https://tools.ietf.org/html/rfc2865 (visited on 08/13/2018).

[15] J. Granjal, E. Monteiro, and J. S. Silva, "Security for the internet of things: A survey of existing protocols and open research issues", *IEEE Communications Surveys Tutorials*, vol. 17, no. 3, pp. 1294–1312, 2015, ISSN: 1553-877X. DOI: 10.1109/COMST.2015.2388550.

[16] B. C. Neuman and T. Ts'o, "Kerberos: An authentication service for computer networks", *IEEE Communications Magazine*, vol. 32, no. 9, pp. 33–38, Sep. 1994, ISSN: 0163-6804. DOI: 10.1109/35.312841.

[17] (). What's shibboleth - shibboleth consortium, [Online]. Available: https://www.shibboleth.net/index/ (visited on 08/15/2018).

[18] *OSI model*, in *Wikipedia*, Page Version ID: 851368768, Jul. 21, 2018. [Online]. Available: https://en.wikipedia.org/w/index.php?title=OSI_model&oldid=851368768 (visited on 08/02/2018).

[19] S. Pearman, J. Thomas, P. E. Naeini, H. Habib, L. Bauer, N. Christin, L. F. Cranor, S. Egelman, and A. Forget, "Let's go in for a closer look: Observing passwords in their natural habitat", ACM Press, 2017, pp. 295–310, ISBN: 978-1-4503-4946-8. DOI: 10.1145/3133956.3133973. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3133956.3133973 (visited on 01/24/2018).

[20] J. Andrić, D. Oreški, and T. Kišasondi, "Analysis of phishing attacks against students", in *2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, May 2016, pp. 1423–1429. DOI: 10.1109/MIPRO.2016.7522363.

[21] S. Sinclair, S. W. Smith, S. Trudeau, M. E. Johnson, and A. Portera, "Information risk in financial institutions: Field study and research roadmap", in *Enterprise Applications and Services in the Finance Industry*, ser. Lecture Notes in Business Information Processing, Springer, Berlin, Heidelberg, Dec. 8, 2007, pp. 165–180, ISBN: 978-3-540-78549-1 978-3-540-78550-7. DOI: 10.1007/978-3-540-78550-7_11. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-540-78550-7_11 (visited on 07/06/2018).

[22] J. Weidman and J. Grossklags, "I like it, but i hate it: Employee perceptions towards an institutional transition to BYOD second-factor authentication", in *Proceedings of the 33rd Annual Computer Security Applications Conference*, ser. ACSAC 2017, New York, NY, USA: ACM, 2017, pp. 212–224, ISBN: 978-1-4503-5345-8. DOI: 10.1145/3134600.3134629. [Online]. Available: http://doi.acm.org/10.1145/3134600.3134629 (visited on 04/10/2018).

[23] *Campus network*, in *Wikipedia*, Aug. 26, 2017. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Campus_network%5C&oldid=797361134 (visited on 05/22/2018).

[24] (). Open source search & analytics · elasticsearch, [Online]. Available: https://www.elastic.co (visited on 08/30/2018).

[25] M. V. Mahoney and P. K. Chan, "An analysis of the 1999 DARPA/lincoln laboratory evaluation data for network anomaly detection", in *Recent Advances in Intrusion Detection*, G. Vigna, C. Kruegel, and E. Jonsson, Eds., red. by G. Goos, J. Hartmanis,

and J. van Leeuwen, vol. 2820, Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 220–237, ISBN: 978-3-540-40878-9 978-3-540-45248-5. DOI: 10.1007/978-3-540-45248-5_13. [Online]. Available: http://link.springer.com/10.1007/978-3-540-45248-5_13 (visited on 08/13/2018).

[26] C. Chen, "Using random forest to learn imbalanced data", University of California, Berkley, Department of Statistics, 666, Jul. 2004, p. 12.

[27] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification (2Nd Edition)*. New York, NY, USA: Wiley-Interscience, 2000, ISBN: 978-0-471-05669-0.

[28] L. Breiman, "Consistency for a simple model of random forests", University of California, Berkley, Department of Statistics, 670, Sep. 9, 2004, p. 10.

[29] R. Caruana and A. Niculescu-Mizil, "An empirical comparison of supervised learning algorithms", ACM Press, 2006, pp. 161–168, ISBN: 978-1-59593-383-6. DOI: 10.1145/1143844.1143865. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1143844.1143865 (visited on 08/07/2018).

[30] C. Bergmeir, R. J. Hyndman, and B. Koo, "A note on the validity of cross-validation for evaluating autoregressive time series prediction", *Computational Statistics & Data Analysis*, vol. 120, pp. 70–83, Apr. 2018, ISSN: 01679473. DOI: 10.1016/j.csda.2017.11.003. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0167947317302384 (visited on 08/07/2018).

[31] C. Cochrane. (May 19, 2018). Time series nested cross-validation, Towards Data Science, [Online]. Available: https://towardsdatascience.com/time-series-nested-cross-validation-76adba623eb9 (visited on 06/23/2018).

[32] (). Fail2ban, [Online]. Available: www.fail2ban.org (visited on 07/02/2018).

[33] Synology. (). Login analysis — synology inc., [Online]. Available: https://www.synology.com/en-global/knowledgebase/DSM/help/DSM/SecurityScan/securityscan_idprotection (visited on 07/01/2018).

[34] Netwrix. (). How to monitor user logons in a domain, Netwrix, [Online]. Available: https://www.netwrix.com/how_to_monitor_user_logons_in_domain.html (visited on 07/05/2018).

[35] Splunk. (). User activity monitoring, Splunk Documentation, [Online]. Available: http://docs.splunk.com/Documentation/ES/5.1.0/User/UserRisk (visited on 07/05/2018).

[36] MaxMind. (). IP geolocation and online fraud prevention, MaxMind, [Online]. Available: https://www.maxmind.com/en/home (visited on 07/05/2018).

[37] A. D. Kent, "Cyber security data sources for dynamic network research", in *Dynamic Networks and Cyber-Security*, ser. Security Science and Technology Volume 1, vol. Volume 1, 0 vols., WORLD SCIENTIFIC (EUROPE), Dec. 3, 2015, pp. 37–65, ISBN: 978-1-78634-074-0. DOI: 10.1142/9781786340757_0002. [Online]. Available: https://www.worldscientific.com/doi/abs/10.1142/9781786340757_0002 (visited on 07/30/2018).

[38] D. M Freeman, S. Jain, M. Duermuth, B. Biggio, and G. Giacinto, "Who are you? a statistical approach to measuring user authenticity", Jan. 1, 2016. DOI: 10.14722/ndss.2016.23240.

[39] E. Shi, Y. Niu, M. Jakobsson, and R. Chow, "Implicit authentication through learning user behavior", in *Proceedings of the 13th International Conference on Information Security*, ser. ISC'10, Berlin, Heidelberg: Springer-Verlag, 2011, pp. 99–113, ISBN: 978-3-642-18177-1. [Online]. Available: http://dl.acm.org/citation.cfm?id=1949317.1949329.

[40] *PostgreSQL 10*. [Online]. Available: https://www.postgresql.org/ (visited on 09/05/2018).

[41] *Timescaledb*, original-date: 2017-03-07T20:03:41Z, Aug. 29, 2018. [Online]. Available: https://github.com/timescale/timescaledb (visited on 08/29/2018).

[42] P. D. Kobezak, "Frequent inventory of network devices for incident response: A data-driven approach to cybersecurity and network operations", p. 125,

[43] F. e. a. Pedregosa, "Scikit-learn machine learning in python", *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, Oct. 2011, ISSN: 1533-7928.

[44] M. M. A. Pritom, C. Li, B. Chu, and X. Niu, "A study on log analysis approaches using sandia dataset", in *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, Jul. 2017, pp. 1–6. DOI: 10.1109/ICCCN.2017.8038522.

[45] D. Pany, *GeoLogonalyzer is a utility to analyze remote access logs for anomalies such as travel feasibility and data center sources*, original-date: 2018-05-18T20:51:07Z, Jun. 25, 2018. [Online]. Available: https://github.com/fireeye/GeoLogonalyzer (visited on 07/01/2018).

# Appendices

# Appendix A

# A IRB Protocol and Approval

This research was conducted under the Virginia Tech Insitutional Review Board (IRB) protocol number 17-375. The following pages are the submitted protocal and approval letter.

VIRGINIA TECH

**Office of Research Compliance**
Institutional Review Board
North End Center, Suite 4120
300 Turner Street NW
Blacksburg, Virginia 24061
540/231-3732 Fax 540/231-0959
email irb@vt.edu
website http://www.irb.vt.edu

**MEMORANDUM**

**DATE:** May 16, 2018

**TO:** David Richard Raymond, Mark Edward DeYoung, Alex Hsu, Philip D Kobezak, Amanda Teh, Ryan Shaune Kingery, Zachary Burch, et. al.

**FROM:** Virginia Tech Institutional Review Board (FWA00000572, expires January 29, 2021)

**PROTOCOL TITLE:** IT Security Office & Lab Data Analytics

**IRB NUMBER:** **17-375**

Effective May 15, 2018, the Virginia Tech Institution Review Board (IRB) approved the Continuing Review request for the above-mentioned research protocol.

This approval provides permission to begin the human subject activities outlined in the IRB-approved protocol and supporting documents.

Plans to deviate from the approved protocol and/or supporting documents must be submitted to the IRB as an amendment request and approved by the IRB prior to the implementation of any changes, regardless of how minor, except where necessary to eliminate apparent immediate hazards to the subjects. Report within 5 business days to the IRB any injuries or other unanticipated or adverse events involving risks or harms to human research subjects or others.

All investigators (listed above) are required to comply with the researcher requirements outlined at:

http://www.irb.vt.edu/pages/responsibilities.htm

(Please review responsibilities before the commencement of your research.)

**PROTOCOL INFORMATION:**

| | |
|---|---|
| Approved As: | **Expedited, under 45 CFR 46.110 category(ies) 5** |
| Protocol Approval Date: | **May 30, 2018** |
| Protocol Expiration Date: | **May 29, 2019** |
| Continuing Review Due Date*: | **May 15, 2019** |

*Date a Continuing Review application is due to the IRB office if human subject activities covered under this protocol, including data analysis, are to continue beyond the Protocol Expiration Date.

**FEDERALLY FUNDED RESEARCH REQUIREMENTS:**

Per federal regulations, 45 CFR 46.103(f), the IRB is required to compare all federally funded grant proposals/work statements to the IRB protocol(s) which cover the human research activities included in the proposal / work statement before funds are released. Note that this requirement does not apply to Exempt and Interim IRB protocols, or grants for which VT is not the primary awardee.

The table on the following page indicates whether grant proposals are related to this IRB protocol, and which of the listed proposals, if any, have been compared to this IRB protocol, if required.

VirginiaTech

**Institutional Review Board**
Existing Data Research Protocol

Note: complete this application only if this research project ***only*** involves the collection or study of **existing data**. Once complete, upload this form as a Word document to the IRB Protocol Management System: https://secure.research.vt.edu/irb

**1. DO ANY OF THE INVESTIGATORS OF THIS PROJECT HAVE A REPORTABLE CONFLICT OF INTEREST?** (http://www.irb.vt.edu/pages/researchers.htm#conflict)

☒ **No**
☐ **Yes,** explain:

**2. IS THIS RESEARCH SPONSORED OR SEEKING SPONSORED FUNDS?**

☒ **No,** go to question 3
☐ **Yes,** answer questions within table

| IF YES |
|---|
| **Provide the name of the sponsor [if NIH, specify department]:** |
| **Is this project receiving or seeking federal funds?**<br>☐ No<br>☐ Yes<br><br>**If yes,**<br><br>**Does the grant application, OSP proposal, or "statement of work" related to this project include activities involving human subjects that are <u>not</u> covered within this IRB application?**<br>☐ No, all human subject activities are covered in this IRB application<br>☐ Yes, however these activities will be covered in future VT IRB applications, these activities include:<br>☐ Yes, however these activities have been covered in past VT IRB applications, the IRB number(s) are as follows:<br>☐ Yes, however these activities have been or will be reviewed by another institution's IRB, the name of this institution is as follows:<br>☐ Other, explain:<br><br>**Is Virginia Tech the primary awardee or the coordinating center of this grant?**<br>☐ No, provide the name of the primary institution:<br>☐ Yes |

**3. DESCRIBE THE BACKGROUND, PURPOSE, AND ANTICIPATED FINDINGS OF THIS STUDY:**

**Co-investigators from the IT Security Office (ITSO) and IT Security Lab (ITSL) will use observational data collected from Virginia Tech's operational network to conduct research in network security data analytics. Personally identifying information will be de-identifed prior to research use. The purpose of this research is:**

**1) Network Security Data Analytics - Develop means to detect and report anomalous behaviours generated**

**by peoples' interactions with information systems. In this area we anticipate findings that enhance network security and improve users situational awareness.**

**2) Privacy Preserving Data Publication (PPDP) - Develop means that support Privacy Preserving Data Publication (PPDP) of de-identified observational data. The two primary avenues of research are anonymization techniques and synthesis techniques that add additional data privacy protections beyond those provided by de-identification processes. We anticipate that we will discover adversarial methods to attack PPDP techniques and develop defensive countermeasures that defeat or delay adversarial attempts to re-identify data.**

**4. EXPLAIN WHAT THE RESEARCH TEAM PLANS TO DO WITH THE STUDY RESULTS:**

*For example - publish or use for dissertation*

**Study results will be published as generalizable knowledge. While we do not have specific venues targeted at this time we plan to publish results in journal articles and conference proceedings.**

**5. WILL PERSONALLY IDENTIFYING STUDY RESULTS OR DATA BE RELEASED TO ANYONE OUTSIDE OF THE RESEARCH TEAM?**

*For example – to the funding agency or outside data analyst, or participants identified in publications with individual consent*

☒ **No**
☐ **Yes,** to whom will identifying data be released?

**6. WILL THE RESEARCH TEAM COLLECT AND/OR BE PROVIDED PARTICIPANT IDENTIFYING INFORMATION (E.G., NAME, CONTACT INFORMATION, VIDEO/AUDIO RECORDINGS)?**

☐ **No,** go to question 7
☒ **Yes,** answer questions within table

| IF YES |
|---|
| **Describe if/how the study will utilize study codes: We will use one-way hashing with a secret password (AKA a hash salt) to generate study codes that correspond to individual persons' identifying information.** |
| **If applicable, where will the key** [i.e., linked code and identifying information document (for instance, John Doe = study ID 001)] **be stored and who will have access?**<br>**The key (mapping between study codes and identifying information) will not be stored. The specific hashing algorithm will not be masked but the secret password will only be known to specific co-investigators (AKA "honest brokers") specified by the principal investigator.**<br><br>*Note: the key should be stored separately from subjects' completed data documents and accessibility should be limited.* |

**7. HOW WILL DATA BE STORED TO ENSURE SECURITY (E.G., PASSWORD PROTECTED COMPUTERS, ENCRYPTION) AND LIMITED ACCESS?**

**Data will be stored on Virginia Tech owned, password protected computers. Data will only be explicitly accessible by co-investigators who have executed a non-disclosure agreement (NDA) with the IT Security Office and completed initial IRB training.**

**8. WHO WILL HAVE ACCESS TO STUDY DATA?**

**Co-investigators who have executed a non-disclousre agreement (NDA) with the IT Security Office and completed initial IRB training. Co-investigators and personel from the IT Security Office will have access to the study data.**

**9. DESCRIBE THE PLANS FOR RETAINING OR DESTROYING STUDY DATA:**

**Data will be retained for no more than 24 months. The start of the time period will be the creation date of the event recorded in observational data. In general, we expect most data to be expired and destroyed within 6 months of collection. Specific data subsets may be retained for longer periods (more than 24 months). In this case the research protocol will be updated to identify the data retained. Data will be destroyed by electronically removing stored data files.**

**10. FROM WHERE DOES THE EXISTING DATA ORIGINATE?**

**Existing data originates from Virginia Techs operational network. As people and information systems interact with network mid-points (servers and network equipment) they produce event data. Some of this data is electronically aggregated in a centralized Log Aggregation & Analysis (LAA) system that is operated by Virginia Tech's Communications Network Services (CNS) and Network Infrastructure & Services (NIS). IT Security Office personel will query data subsets from the LAA system and then apply de-identification procdures.**

**11. PROVIDE A DETAILED DESCRIPTION OF THE EXISTING DATA:**

**The existing observational data contains the following directly identifying data elements:**
**- user names (the user's VT PID or Hokies user name)**
**- user e-mail addresses**

**The identifying data elements will be de-identified by IT Security Office personel before the data is made availble to IT Security Lab researchers.**

**Other data elments (that are not identifying) include:**
**- End-point device Media Access Codes (MACs) which are associated with users devices when they connect to network resources**
**- Internet Protocol (IP) numbers which are associated with the MAC of a device and a network session**
**- Informational, warning, and error messages about the network resources and users interaction with the network resources**

**12. IS THE SOURCE OF THE DATA PUBLIC?**

☒ **No,** go to question 13
☐ **Yes,** you are finished with this application

**13. WILL ANY INDIVIDUAL ASSOCIATED WITH THIS PROJECT (INTERNAL OR EXTERNAL) HAVE ACCESS TO OR BE PROVIDED WITH EXISTING DATA CONTAINING INFORMATION WHICH WOULD ENABLE THE IDENTIFICATION OF SUBJECTS:**

- **Directly** (e.g., by name, phone number, address, email address, social security number, student ID number), or

- **Indirectly through study codes** even if the researcher or research team does not have access to the master list linking study codes to identifiable information such as name, student ID number, etc

or

- **Indirectly through the use of information that could reasonably be used in combination to identify an individual** (e.g., demographics)

☒ **No,** collected/analyzed data will be completely de-identified
☐ **Yes,**

**If yes,**

*Research will not qualify for exempt review; therefore, if feasible, written consent must be obtained from individuals whose data will be collected / analyzed, unless this requirement is waived by the IRB.*

**Will written/signed or verbal consent be obtained from participants prior to the analysis of collected data?**
-select one-

***This research protocol represents a contract between all research personnel associated with the project, the University, and federal government; therefore, must be followed accordingly and kept current.***

***Proposed modifications must be approved by the IRB prior to implementation except where necessary to eliminate apparent immediate hazards to the human subjects.***

***Do not begin human subjects activities until you receive an IRB approval letter via email.***

***It is the Principal Investigator's responsibility to ensure all members of the research team who collect or handle human subjects data have completed human subjects protection training prior to handling or collecting the data.***

**----------END----------**

# Appendix B

# B Data Sources

The FIND dataset is a subset fields from a variety of log sources from the CLS extracted between 13 November 2017 and 4 January 2018 (53 days). It has been extracted according to VT IRB #17-375 and **can only be accessed by approved co-investigators**. If you want/need to access this dataset, please request approval.

There are currently eight main types of collected log data available for use, which provide state information for various levels of the VT campus network infrastructure from the application to the physical layer. The breakdown of types, subtypes, descriptions, and counts is shown in table B.1. Below, a detailed breakdown of information about the data, along with missing data points and known inconsistencies is given. The ordering of the data is a breakdown from highest level to lowest level.

## B.1 Login

These logs come from a user on the network accessing an online service through a web browser that needs central authentication. The user is redirected to login.vt.edu. There are three kinds of logs: Duo, LDAP, and Single Sign On (SSO). The log provides a timestamp, requesting IP, and the de-identified PID of the user. SSO also provdies the target URL being authenticated to.

| Type | Subtype | Description | Count |
|---|---|---|---|
| ARGUS | Netflow | Internet Traffic | N/A |
| Login | Shibboleth Duo | Service Access | 1,746,960 |
| | Shibboleth LDAP | | 2,019,918 |
| | SSO Single Sign On | | 5,237,713 |
| DHCPD | Commit | IP to MAC connection | 95,823,812 |
| | Expire | | 5,495,161 |
| | Release | | 10,455,749 |
| NetRecon | IPv4 to MAC | IP to MAC polling | 520,664,135 |
| | IPv6 to MAC | | 456,419,635 |
| | MAC to Port | MAC to Building/Room | 448,601,900 |
| | Port to Outlet | | 7,138,301 |
| RADIUSd | | MAC and IP to User and Sector | 84,972,254 |
| Syslog | Association | MAC to AP | 113,037,931 |
| | Authentication | | 72,342,768 |
| | Auth-Inner | | 8,634,143 |
| | De-authentication | | 8,472,123 |
| | Roam | | 68,837,619 |
| TMM | Allocated | External Port to Internal IP | 11,375,399 |
| | Released | | 11,395,328 |
| VPN | Auth | Outside IP to Internal IP | 80,932 |
| | Tunnel Start | | 83,537 |
| | Tunnel End | | 83,537 |

Table B.1: Table breakdown of collected FIND data.

The Duo subtype occurs whenever a user does a two factor authentication to a computer. Virginia Tech requires a two factor for the first time a computer is used or if two factor hasn't been done in a week. LDAP, Lightweight Directory Access Protocol, is whenever a permissions lookup needs to be done. This occurs when a user authenticates to something that might be restricted. A good example is the CNS Kibana instance, log.it.vt.edu, where the CLS is viewed through. Only a certain group of people are allowed to use that system. Conversely, everyone can use Canvas. Last is the SSO, which triggers whenever a user accesses a URL and needs to authenticate to a backend webserver for a session token. So, the first time Canvas or Hokie Spa is accessed in the day triggers an SSO login.

## B.2 DHCPd

These logs come from the Dynamic Host Configuration Protocol daemon (DHCPd). They provide state change information about what devices on campus have what address. This is only for IPv4, since VT uses SLAAC for IPv6. This information is state change based, thus it's three kinds of logs are for when the device has moved to the subtype's state. All logs contain a timestamp and source IP address. The commit subtype is when a device has requested an IP and

## B.3 Wireless Assosications

## B.4 VPN