Detecting Lateral Movement

An Ensemble Learning and Data Visualization Approach

A Major Qualifying Project submitted to the faculty of Worcester Polytechnic Institute in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science

By

Madeleine Longo Jeffrey Martin Ian Vossoughi

October 12, 2017

Sponsored By

Massachusetts Institute of Technology Lincoln Laboratory

Advised By

Steven Gomez (MIT LL) George Heineman (WPI) Diane Staheli (MIT LL)

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited. This material is based upon work supported by the Department of Defense under Air Force Contract No. FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Department of Defense.

Abstract

In this Major Qualifying Project, we explored utilizing ensemble learning and data visualization to detect lateral movement from Advanced Persistent Threats (APTs) in enterprise networks. We developed a detection framework for analysts to pinpoint malicious events within a cybersecurity dataset from Los Alamos National Laboratory. Our project produced two primary findings: ensemble learning significantly improved the detection rate of malicious events, and a heatmap visualization can provide promising indications of suspicious activity, but remains ultimately insufficient for reliably identifying APTs.

Authorship

The individual contributions with respect to research, design, implementation, and writing were as follows

Research

1. APTs - *Ian Vossoughi*
2. Intrusion Detection - *Jeff Martin*
3. Machine Learning Techniques - *Madeleine Longo & Jeff Martin*
4. Ensemble Techniques - *Madeleine Longo*
5. Data Visualization Techniques - *Ian Vossoughi*
6. Cybersecurity Datasets - *Jeff Martin*
7. Related Work - *Madeleine Longo, Jeff Martin, & Ian Vossoughi*

Writing

1. Introduction - *Madeleine Longo, Jeff Martin, & Ian Vossoughi*
2. Background - *Madeleine Longo, Jeff Martin, & Ian Vossoughi*
3. Methodology - *Madeleine Longo, Jeff Martin, & Ian Vossoughi*
4. Framework - *Jeff Martin*
5. Results - *Madeleine Longo, Jeff Martin, & Ian Vossoughi*
6. Discussion - *Madeleine Longo, Jeff Martin, & Ian Vossoughi*
7. Executive Summary - *Ian Vossoughi*
8. Appendices - *Madeleine Longo & Ian Vossoughi*
9. Formatting - *Madeleine Longo & Jeff Martin*

Design

1. Database Architecture - *Jeff Martin*
2. Framework Components - *Madeleine Longo, Jeff Martin, & Ian Vossoughi*
3. Visualization - *Ian Vossoughi*

Implementation

1. Framework Controller - *Jeff Martin*
2. Los Alamos Dataset Preprocessing - *Madeleine Longo & Jeff Martin*
3. Database Initialization Engine - *Jeff Martin*
4. Database Insertion Engine - *Jeff Martin*
5. Aggregation Engine - *Jeff Martin*
6. Partition Engine - *Jeff Martin*
7. Detector Insertion Engine - *Jeff Martin*
8. Detector Engine - *Jeff Martin*
9. Anomaly Detection Modules (Neural Network, Random Forest, Oracle, & Imperfect Oracle) - *Jeff Martin*
10. Ensemble Creation Engine - *Jeff Martin*
11. Ensemble Engine - *Madeleine Longo*
12. Metrics Engine - *Madeleine Longo*
13. Visualization Backend - *Ian Vossoughi*
14. Visualization Frontend - *Ian Vossoughi*

Acknowledgements

This project's success is in large part a result of the support received from a number of individuals and organizations.

First, we would like to thank WPI Professor George Heineman for coordinating the project center at MIT Lincoln Laboratory, as well as for advising our MQP and challenging us to produce a novel and successful project.

Additionally, we would like to thank MIT Lincoln Laboratory for providing us with the resources and environment needed to complete this project. We greatly appreciate the support we received from our MIT Lincoln Laboratory mentors, Steven Gomez and Diane Staheli. Finally, we would like to thank the cybersecurity analysts and researchers who helped direct and evaluate this project.

Executive Summary

A new class of cyberattack known as the Advanced Persistent Threat (APT) has greatly increased the difficulty of defending networks against intrusions and information theft, mainly due to its stealthy nature and ability to hide within normal network activity. Many organizations use networks with strong external defenses, but significantly weaker defenses for intra-network communications. This vulnerability allows any attackers who gain an initial foothold in the network to spread rapidly via lateral movement between computers. Without an effective method to detect malicious network access, cybersecurity analysts are unable to determine whether an APT is in the network or if their data is at risk of theft.

Due to the abundance of network data events, a small fraction of which represent malicious activity, even relatively accurate anomaly detectors will typically have high false positive rates (causing them to flag a disproportionately large number of events as potentially suspicious). Sorting through each flagged event and performing the necessary follow-up requires significant time and concentration on the part of the analyst, resulting in increased fatigue.

Our goal was to identify effective methods of assisting analysts with the detection of lateral movement, and then develop a proof-of-concept framework to support the analysts with the methods we found. Our framework can ingest large activity records, activate multiple detectors to identify anomalies in the data, and then use methods inspired by ensemble learning (a common strategy of machine learning) to intelligently combine the results from each detector. We hypothesized that by integrating results from multiple detectors on the same data to mitigate the individual weaknesses of each detector, we would improve our framework's overall detection rate while reducing false positives. Furthermore, we planned a visualization based on mapping network events according to time and location (e.g., username, host computer, IP address), which we hypothesized would display clearly anomalous patterns in the event of lateral movement. Analysts could subsequently leverage the framework as a useful first step in their investigation of APTs before looking further into suspicious users or computers. Our framework provides an extensible base for analysts to freely add custom datasets and anomaly detectors, with each component dedicated to independently inserting or handling data in the central database.

We evaluated our framework's effectiveness at reducing false positive rates by calculating the Area Under the Receiver Operating Characteristic Curve (AUROCC), which compares the false positive and true positive rates in a single metric, for each anomaly detector and their combined results. Additionally, we obtained iterative feedback from MIT Lincoln Laboratory (Lincoln Lab) analysts on the effectiveness of our visualization for identifying potential lateral

movement as we improved its design. Finally, we conducted an informal assessment of the framework's usability with analysts and other Lincoln Lab researchers to obtain constructive feedback for how the framework could be improved in the future. During this assessment, we learned more about APTs and potential methods for lateral movement detection.

We evaluated our framework and tested our hypothesis by leveraging Los Alamos National Laboratory's cyber security dataset. We initially implemented two machine learning threat detectors and tested them independently as well as part of an ensemble. To account for the sheer size of the Los Alamos dataset, we trained and tested on a subset of the data. Due to the low AUROCC of one of the detectors, however, our initial results were inconclusive. We repeated the experiment with an artificial detector guaranteed to perform with a high AUROCC. The results of this revised experiment showed that the ensemble detected threats with a significantly higher AUROCC than either component threat detector, demonstrating strong support for our first hypothesis.

For the evaluation of our second hypothesis, we generated an artificially small subset of the Los Alamos dataset and provided analysts with the opportunity to use our visualization with the goal of detecting users performing malicious lateral movement. This small dataset proved necessary since evidence of lateral movement had been virtually impossible to find in visualizations of the large datasets we used to evaluate our first hypothesis. Despite benefiting from an initial training period and the opportunity to ask questions at any point during the evaluations, none of the analysts performed significantly better than random guessing. As such, the results from our experiment do not show support for our second hypothesis, resulting in the conclusion that authentication visualizations with time and location values alone are insufficient for extracting lateral movement patterns. However, the visualization excelled at directing analysts toward users with abnormal numbers of accessed workstations, and it can therefore function as an initial guiding step for highlighting potentially suspicious users.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Organizations that rely on computers to work on sensitive data have always been at risk of data theft due to the inherently vulnerable nature of hardware and software. Traditionally, cyberattacks (malicious actions in cyberspace) have used any and all resources at their disposal to gain disruptive access to as many computers as possible, stealing and selling any valuable information uncovered in the chaos. This changed in the early 2000s, when advances in anti-malware technology and vulnerability patching made such "smash and grab" attacks unrealistic to successfully execute. In response, attackers have switched tactics, focusing on methodical exfiltration operations over sweeping and destructive cyberattacks. Computer networks (and any sensitive data they contain) are now extremely vulnerable to specialized attacks aimed at specific targets. This has been demonstrated with a series of large-scale cyberattacks, including Titan Rain (2003), Operation Aurora (2009), Stuxnet (2010), and Red October (2012). These cyberattacks share a number of common features: they targeted specific organizations, remained persistent throughout the attack, and utilized advanced exploits. These three characteristics define a class of cyberattacks known as Advanced Persistent Threats (APTs) [13].

APTs generally follow a consistent set of steps to complete their infiltration and information theft operations. The steps of a typical APT are listed below, with the progress of a hypothetical APT described in more detail at each step [13]:

1. **Reconnaissance and Weaponization:** Plan the attack by gathering information about the target organization through social engineering and open-source intelligence tools.
    (a) **Example:** The APT attacker, after examining the social media accounts of known low-level employees in the organization, finds that these employees are frequently overworked and pressured by their supervisor to quickly address any emails he sends them, including fully reading attached reports or presentations.
2. **Delivery:** Provide the exploit to the target, either directly (i.e., with phishing) or indirectly (i.e., with 'watering hole' attacks in which commonly-used resources are compromised).

(a) **Example:** The APT spoofs their email address to closely resemble that of the aforementioned supervisor. The attacker then sends emails with the subject line "**URGENT** Report Review" and includes a PDF containing malicious JavaScript code.

3. **Initial Intrusion:** Gain a "foothold" (access to a single host) in the network via either stolen credentials or malicious code executed through a vulnerability in the victim's software.

   (a) **Example:** Once the employees receive the email, they quickly open the report to comply with the demand of their "supervisor". The PDF report appears legitimate, but after an employee agrees to run JavaScript on the form, the script executes malware to start remote access tools for the APT to gain access to the employee's computer.

4. **Command and Control:** Control compromised computers to begin exploiting the network. This step usually remains subtle and uses legitimate services, which are not usually detected by anti-malware tools.

   (a) **Example:** The APT gains a foothold in the network by using the executed remote access tools to log into the compromised computer.

5. **Lateral Movement:** Move through the network by stealing the credentials of other users (ideally gaining administrator or IT personnel information for maximum control) in search of valuable data.

   (a) **Example:** The APT downloads and runs tools designed to escalate their new user privileges and extract stored password hashes on the compromised computer. In the process, they find a Virtual Machine (VM) that many users have logged onto and use their acquired password hashes to log into the VM itself. From here, the APT repeats the privilege-escalation and credential theft to log onto computers with credentials accessible from the VM.

6. **Data Exfiltration:** Compress and encrypt acquired sensitive data before transmitting it out of the network with a secure protocol.

   (a) **Example:** One of the VM users holds employee payroll spreadsheets in their desktop active directory, which demonstrates that the organization has been violating minimum wage laws. Once the APT finds this data, they use the Secure Copy Protocol (SCP) to copy the payroll spreadsheet to a secure computer outside the network, then proceed to submit these records to WikiLeaks in order to expose the organization's private data.

Despite what this simplified example indicates, the APT does not necessarily exit the network after its initial data exfiltration. APTs can linger in networks for months or even years without being detected, repeatedly exfiltrating data and building network backdoors for future infiltrations. There are several ways to mitigate the effects of APTs before they have reached their target. The first and most frequently used method is prevention, or securing an organization's cyberspace against external threats to deter all but the most persistent

APTs. The second is detection, or analyzing events in an organization's cyberspace for signs of APTs. The third, initiating an appropriate response to the detected APT, is vital for mitigating further network damage during an ongoing threat, but will produce limited results if the first two steps provided insufficient protection against APTs [13]. In this Major Qualifying Project (MQP), we focus on supporting cybersecurity analysts (specialists in preventing, detecting, and responding to threats) with the detection of the lateral movement stage of APTs in enterprise networks.

Problem Statement

Detecting lateral movement is challenging because APTs generally move through the network extremely slowly and subtly. Though their behavior differs from that of a normal user or administrator, these differences often occur in ways that are difficult to detect, especially since anti-malware tools completely ignore most APT attack vectors [13].

Additionally, many organizations have large networks that generate far too many events for any single analyst to inspect manually [13]. Thus, analysts will frequently conduct intrusion detection autonomously through either signature-based or anomaly-based detection systems. Although such automated approaches to intrusion detection have demonstrated an ability to label malicious events with a relatively low false negative rate, they often incur high false positive rates.

This MQP addresses the following two problems with lateral movement detection:

1. **Problem One:** Automated intrusion detectors often have high false positive rates.
2. **Problem Two:** Cybersecurity analysts cannot easily distinguish lateral movement from normal behavior.

Hypotheses

We drew motivation from data visualization and ensemble learning, respectively, to solve the two problems stated in the Problem Statement. Data visualization is the practice of graphically presenting data to an analyst, who can learn patterns that are difficult or impossible for automated systems to detect. Ensemble learning intelligently combines multiple biased predictions (which are synonymous with alerts in this context) into a single, more accurate prediction. They are typically used in machine learning, where the strategy has been shown to produce models with an improved Area Under the Receiver Operating Characteristic Curve (AUROCC) (a metric that accounts for both the false positive rate and the false negative rate of the prediction). We predicted that ensembles would help address the first problem specified by this MQP and data visualization would help address the second. More specifically, our MQP posed the following hypotheses:

1. **Hypothesis One:** An anomaly detector ensemble that combines the predictions from a sufficiently diverse set of component anomaly detectors will detect malicious events

with an AUROCC that is greater than or equal to the average AUROCC of its component detectors.

2. **Hypothesis Two:** A visualization that organizes anomalous events based on each event's time, location, and suspicion level will allow an analyst to determine whether a given pattern of events is caused by lateral movement.

Goals

Because our MQP is hypothesis-driven, we primarily focused on determining the validity of both hypotheses listed above. Additionally, we developed a proof-of-concept framework to help analysts detect lateral movement by providing the following features:

- Supporting the insertion of any time-stamped cybersecurity dataset that is location-oriented (i.e., has a field that specifies *where* the event occurred) and includes ground truth (a method to reliably determine which events are malicious over some subset of the data).
- Supporting the installation of anomaly detectors in a language-agnostic plugin system.
- Assigning events with a suspicion level (predicted probability of the given event being malicious) based on an ensemble of installed anomaly detectors.
- Creating an interactive visualization of events based on their time, location, and suspicion level.

We designed our framework to accept arbitrary datasets and anomaly detection plugins as well as a variety of visualization strategies. Next, we developed our sample detectors as simple machine learning models, and we built our proof-of-concept visualization as a heatmap based on the events' time, location (i.e., the user or computer associated with the event), and suspicion level. We decided to limit our project to visualizations using these three dimensions because they would be the most useful for detecting lateral movement. Because our framework could use machine learning models as detectors, each detector required a clear definition on how it would be trained. We partitioned each dataset was into "train" and "test" data, and only showed the "test" data in our visualization. This practice ensured that all machine learning models could train on the same portion of the data and thus could support meaningful comparisons between each other.

After designing an initial structure, we conducted interviews with analysts to incorporate their input into our design. We determined that our visualization would enable a set of basic user interactions: adding and managing datasets and detector plugins, filtering events, changing the field used for location, and selecting a specific cell to see all events that occurred in that combination of location and timeframe. We decided not to implement capabilities such as note-taking, a highly useful feature for analysts [10], because pure usability components lay outside the scope of our project and could easily be implemented in future versions of our framework. To the best of our knowledge, a framework using the same backend and

visualization concepts had not yet been attempted; our overall goal was to test its potential for future use and development.

Evaluation

We evaluated our framework based on its performance for the required use cases and how thoroughly it enabled us to prove or disprove our hypotheses. We would designate our framework as highly successful if it validated our hypotheses and proved usable and stable during experimental usage, but disproving our hypotheses would not automatically invalidate the framework's value. Evidence against our hypotheses would indicate that our proof-of-concept visualization design was insufficient for detecting lateral movement, providing an informative result for lateral movement detection research as a whole.

We evaluated Hypothesis One with two detectors, both implemented as simple machine learning models so we could quickly gain accurate predictions, and assessed the result of combining their predictions over multiple trials. We initiated the trial runs by partitioning the data in a reproducible way, training the detectors on a training set, and evaluating their results on the test set. Afterward, we combined the average results across all trials to form the overall result. If the detectors were too similar, or their AUROCC too low, we would repeat the experiment. If the ensemble of detectors had a greater AUROCC than the individual detectors, we would subsequently consider the experiment as validating evidence for Hypothesis One.

We evaluated Hypothesis Two by visualizing the authentication data for a small sample of users over a specific time window. We presented our visualization to multiple analysts and asked them to determine which users in a small dataset were involved in lateral movement. If the analysts found the framework useful and could usually locate lateral movement accurately, the experiment would serve as evidence toward Hypothesis Two. Conversely, if the analysts could not reliably detect lateral movement, our experiment would provide evidence against Hypothesis Two.

Chapter 2

Background

2.1 Advanced Persistent Threats

Among the most common modern threats to the security of vital information for companies, governments, and organizations alike, perhaps none have caused such a devastating impact as the cyberattacks classified as Advanced Persistent Threats, or APTs. APTs are advanced because they use highly complex and intricate tactics to enter and navigate a target network, and persistent for their ability to remain undetected once inside a network for long durations of time [46]. From the 2003 Titan Rain attack on FBI and NASA-controlled military data to the 2010 Stuxnet Worm designed to steal nuclear infrastructure data from Iran, APTs continue to remain the threat model of choice for large, sophisticated organizations in search of heavily-guarded intellectual property [46].

There are several factors that make APTs critically damaging in nature: in particular, unlike traditional attacks aimed at simply spreading damage across their victim's domain and stealing any information available, an APT will virtually always have a specific target in mind when conducting reconnaissance and will search for exceptionally valuable data to acquire (e.g., national security data, trade secrets) [13]. This leads to measured, premeditated attacks upon the target organization that are specifically designed to exploit the target network's vulnerabilities, frequently utilizing tailor-made social engineering tactics for users in the network [46]. In addition, because APTs normally aim for a very narrow set of intellectual property, the perpetrators are likely to organize long-term campaigns and persist in their operations over a far longer span of time than other attackers would [13]. As a result, APTs are considerably less likely to be deterred or shift their focus if initial break-in attempts are unsuccessful. Finally, due to the delicate nature of their operations, APTs usually prioritize stealth when orchestrating any illicit activity in the target network [13]. APTs will use tactics such as encryption of network traffic to hide themselves within normal user activity as long as possible, sometimes remaining undetected until months after their attack has concluded [13]. With an emphasis on relentlessly pursuing specific targets and attempting to avoid detection for as long as possible, APTs pose security challenges that deviate widely from the norm.

Lateral Movement

For any APT requiring information that is inaccessible from the initial compromised computer, lateral movement is essential to gain access to high-value information in the network. Once the APT obtains complete access to the foothold computer (the first computer the APT reaches in the network), they will attempt to progress into more network devices, primarily by utilizing resources available on their foothold computer. By accessing restricted files and stealing other users' credentials with privilege escalation attacks, the APT can begin logging into new computers and accounts, expanding their hold on the network. This campaign of credential theft enables APTs to masquerade as a number of legitimate users within the network, including some with increased privileges such as administrators or IT personnel [37] [47]. This pattern of attack allows attackers to gradually expand across hosts in the network, increasing their level of control while accumulating sensitive information [15].

Strategies for Executing Lateral Movement

Since attackers require both user credentials and privilege escalation to begin accessing other network computers, many lateral movement methods involve delving into the computers the APT controls to find any data that allows access to other users. One of the most effective methods of obtaining new user credentials is a pass-the-hash exploit to harvest cached password hashes on user computers. In order to allow legitimate users to remotely access network resources, Windows computers on a network will store the results of their passwords locally after entering a hash function; authenticating users who have not yet accessed the remote network will have their passwords compared against this local hash to gain access [37]. While this process of saving password hashes locally provides a simple method of authentication for users logging in remotely, it also provides an avenue of approach for APTs to access the password hashes and use them to authenticate as various other users [37]. Similarly, attackers can copy Kerberos Ticket Granting Tickets (TGTs) provided to users after successful logins for repeated access to network resources in a pass-the-ticket exploit [44].

Pass-the-hash attacks are primarily possible due to inherent vulnerabilities present within MS Windows' Single Sign-On (SSO) authentication systems, including both the archaic LAN Manager (LM) and current NT LAN Manager (NTLM) methods [14]. When a user successfully logs into a Windows-based network by authenticating toward a domain controller on a client computer, the Local Security Authority Subsystem Service (LSASS) process will store the password hash in memory on this computer [22]. After the initial login and hash storage, the user can access permitted network resources without repeatedly entering their password; on each attempt, the network server will issue a login challenge request to the client, which will automatically present their local password hash and request for the domain controller to compare it against the server's copy of the hash [38]. If the two hashes match, the server will be approved to complete the client's original request. As such, under the Windows LM or NTLM authentication methods, an attacker who has obtained a user's password hash will receive access to the same network resources as those available from

having obtained a cleartext password. Figure 2.1 illustrates the usage of password hashes in a pass-the-hash attack [14].

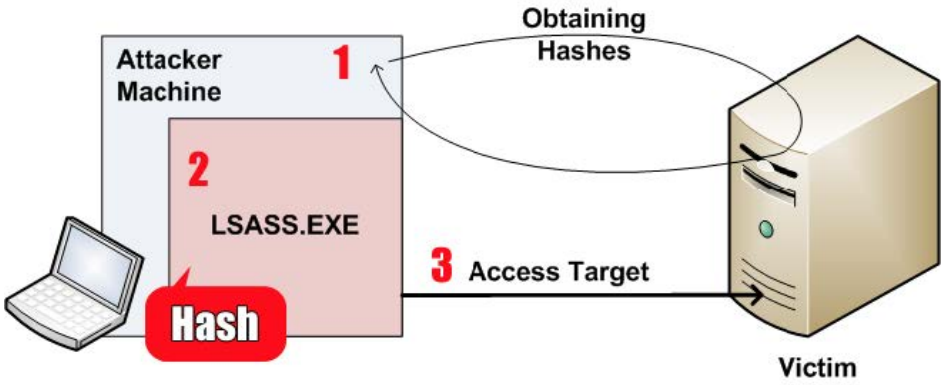


Figure 2.1: Complete illustration of a pass-the-hash attack using password hashes, courtesy of Ed Skoudis [14].

The Internet provides a multitude of free tools to facilitate hash dumping, and while these tools require administrative rights on the target computer to dump hashes from Windows databases or memory, they provide a straightforward method of initiating a pass-the-hash attack once this prerequisite has been met. Local tools such as *pwdump*, *fgdump*, and *gsecdump* use DLL injection techniques to obtain hashes from active and cached login sessions, providing a quick method of accessing hashes for later use if the attacker receives local access to their compromised computer [14]. However, remotely dumping hashes is also a possibility; after establishing a connection with the target computer, attackers can combine the aforementioned local tools with Windows PsExec (a system tool to execute processes on remote machines) to begin the hash collection process remotely [22]. Furthermore, several pass-the-hash attack suites, including Metasploit and Hernan Ochoa's Pass-The-Hash Toolkit, combine the initial hash dump process with tools to begin directly utilizing newly collected hashes for authentication, streamlining the attack process for the APT's benefit [22].

Made possible by inherent Windows authentication vulnerabilities and exacerbated by well-supported online tools, pass-the-hash remains an extremely successful method for APTs to obtain stolen credentials and propagate through their target networks. Since authentications with password hashes can be either a benign operation performed by remote network users or a malicious act of lateral movement started by attackers, the mere use of hashes cannot be used to cleanly differentiate the two or trace APTs by their login records alone [38]. Instead, analysts viewing authentication records must search for notable behavioral anomalies pertaining to the users who have logged in during the recorded time range. There is no single algorithm to conclusively designate anomalous user behavior as indicative of lateral movement, but common heuristics include significant deviations from the user's normal work schedule (i.e., logging in at 3:00 AM on a Saturday morning), multiple new network machine logins in a short period of time, or attempts to access network resources or devices normally above their privilege level. By investigating users who exhibit these behaviors in their network logs, analysts can eliminate users with justified reasons for performing the above and search for further anomalous activity in other logs (e.g., processes, DNS) from those who

remain.

Lateral Movement Detection

Since an APT will increase their chances of obtaining sensitive data each time they gain access to a new computer, lateral movement functions as a critical step for an APT; its level of success will heavily dictate the impact of the entire campaign, along with the potential success for future APTs if the attackers install a network backdoor after obtaining administrative access in the network [47]. Lateral movement is also frequently the longest and most cautious step within an entire APT's campaign. APTs normally camouflage their movements and activities among benign network traffic to remain undetected. By avoiding any actions that would appear immediately suspicious to monitoring analysts, the APT gradually expands their reach in the network [13]. This makes detecting and responding to lateral movement as early as possible a critical step in minimizing the overall damage an attacker can cause.

However, introducing measures to either prevent lateral movement or detect an in-progress attack can be extremely challenging or impractical for network analysts. While pass-the-hash and pass-the-ticket exploits can be prevented with multi-factor authentication (requiring the user to verify their identity on a second secure device rather than simply providing the password hash), they remain some of the most frequently utilized methods of gaining control over a computer [37]. Since these methods will generally use legitimate administrative tools (e.g., PsExec) or other authorized methods to cover their tracks, analysts cannot receive explicit alerts regarding whether password hashes have been exploited during an authentication event [47]. Rather than build monitoring tools to directly detect a pass-the-hash attack, analysts must generally investigate the authentication records collected on the network in hopes of discovering unusual patterns of user activity [44].

2.2 Intrusion Detection

As defined by the National Institute of Standards and Technology (NIST), intrusion detection is "the process of monitoring the events occurring in a computer system or network and analyzing them for signs of possible incidents" [42]. However, due to the number of events under analysis, the problem is impractical for human consideration alone [7]: as shown in a report on intrusion detection, manual detection only found 2% of the malicious events that an automated system detected [8]. Intrusion detection is often automated using either signature-based or anomaly-based approaches, as explained below [8] [48] [26].

Signature-Based Approaches

Signature-based approaches (also referred to as misuse-based approaches) classify events as either *malicious* or *benign* based on "signatures", or patterns of events that are known to be malicious. For example, events related to creating symbolic links to password files in UNIX

systems would be flagged by these approaches, as such activity is frequently correlated with privilege escalation attacks. In general, signature-based approaches are appropriate when reliable detection of known attacks is necessary; however, they cannot detect novel attack methods [8] [26].

Signature-based approaches may be implemented in one of two ways: expert systems and state transitions. Expert systems allow analysts to enter classification rules, typically as if-else statements, that determine how the engine will classify events. Doing so allows the analyst to control the performance of the classification engine [8]. One such expert system is the APT-Hunter [44], which we will discuss in detail in Section 2.6: Related Works. Conversely, state transition detection models the states of the network and the transitions between these states to detect intrusions. This allows state transition models to easily detect variants of known attacks, as well as slow attacks that are distributed throughout the network [8].

Anomaly-Based Approaches

Anomaly-based approaches flag an event as *anomalous* (or *malicious*) if the event deviates from what is known to be *normal* (or *benign*) behavior [8] [26] [48]. Anomaly-based intrusion detection relies on two basic assumptions:

- Users exhibit patterns of behavior that are both predictable and consistent [8].
- The behavior of an attacker differs from that of a normal user [26].

In general, anomaly-based approaches have the advantage of detecting new forms of attacks; however, they often suffer from high false positive rates [26]. Although anomaly-based intrusion detection can be realized without machine learning [8], the field is mainly dominated by machine learning implementations [48].

Machine Learning for Anomaly Detection

Machine learning is the practice of developing algorithms that learn to make intelligent predictions based on past data. Machine learning has been widely applied to intrusion detection because it allows for flexibility and adaptability when modeling interdependencies. However, supervised machine learning requires a set of labeled training data, which is often difficult to obtain in the domain of cybersecurity [48].

Common machine learning techniques that have been applied to anomaly-based intrusion detection include the following [48]:

- **Bayesian Network:** A model that represents the conditional dependencies between random variables through a directed graph.
- **Markov Model:** A model that represents the system as a set of visible (Markov Chain) or hidden (Hidden Markov Model) states and transitions.

- **Neural Network:** A model that make decisions by propagating data through a network of neurons, each of which has an input, output, and activation function.
- **Fuzzy Logic Techniques:** A technique that uses approximate reasoning from fuzzy set theory to make predictions.
- **Genetic Algorithms:** A technique based on ideas from evolutionary biology (e.g., recombination, selection, inheritance, and mutation) to generate a model over a series of 'generations' with iterative improvements.
- **Clustering and Outlier Detection:** A technique that uses distance measurements to form clusters of similar data points.
- **Data Mining:** A technique using sophisticated algorithms to find patterns and relationships in large datasets.

The performance of a classifier is generally measured in terms of the Receiver Operating Characteristic (ROC) curve. A ROC curve "depicts the relative trade off between the benefits (True Positives) and the costs (False Positives)" [16] by plotting the true positive rate against the false positive rate (at different cutoff points for models that give continuous output, or at a single point for models that give discrete classifications). Figure 2.2 demonstrates a sample ROC curve. In this MQP, we will evaluate the performance of a machine learning model by calculating the Area Under the Receiver Operating Characteristic Curve (AUROCC), which is an important and commonly-used metric in machine learning [16].

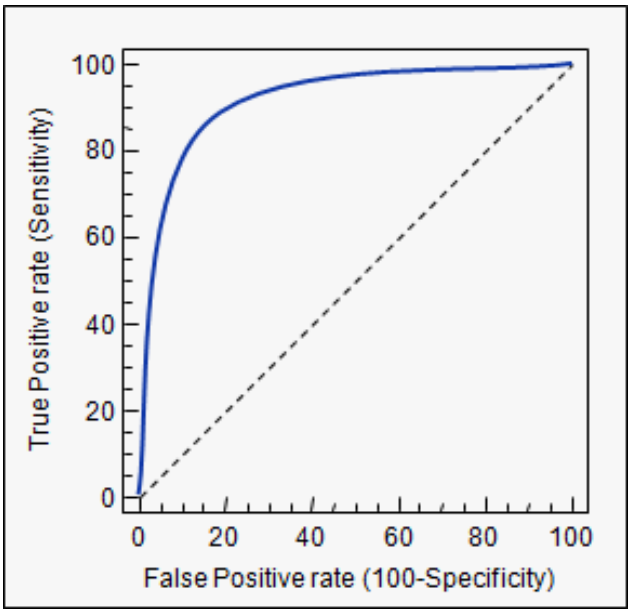


Figure 2.2: An example Receiver Operating Characteristic (ROC) curve. The solid line represents the performance of the actual classifier, and the dashed line represents the performance from random guessing [4].

2.3 Ensemble Predictions

Ensemble learning is an increasingly popular technique to improve the results of machine learning models. It uses a diverse set of models to achieve results better than those of any individual model. Although the algorithms used to combine predictions are mainly used in the context of machine learning, several of them can be used in any context where multiple predictions are combined into a single result.

The intuitive explanation behind ensemble learning states that if each model has different errors, and the results of all models are combined intelligently, the overall error rate will subsequently drop [40]. The mathematics of ensemble learning is outside of the scope of this project, but can be found in articles by Ludmila [30], Polikar [40], and Krogh [45]. At worst, ensemble learning simply reduces the risk of overfitting, which refers to the tendency of models to learn details specific to the training set and thus lower their ability to generalize on real data. Although the accuracy of the ensemble may not exceed that of its most accurate model, the ensemble will be much less likely to make egregious errors. However, in many cases, ensemble learning can approach the optimal error rate as long as the error rate of each individual model is less than 50% [40].

ensemble learning may also be appliable for problems that are difficult or impossible to solve with a single model. This can happen if the dataset is too large, the problem is too complex, or the data comes from multiple sources (or is otherwise heterogeneous). Ensemble learning allows each model to focus on a manageable subset of the problem space, and an intelligent rule for combining them frequently makes the ensemble more efficient and effective than any one model [40].

In theory, large and well-designed ensembles can achieve globally optimal performance in the general case [45], which is better than can be expected of any one model. Although ensembles of such size are not feasible to implement, realistically sized ensembles can still achieve good results. Real data is noisy, but ensembles are designed to deal with noise. Training individual models on subsets of the data results in biased models, but their biases cancel out if they have sufficient differences, measured as "diversity" [45]. Ensemble diversity is still ill-defined [30], but it can be increased by creating models that differ as much as possible from each other. This is performed by training different types of models, training each model on different datasets, or using different features to train different models. An especially effective way to ensure ensemble diversity is to use "unstable" models, which are sensitive to small differences in their training data and will thus diverge even on similar data. Unstable models are typically employed on different (yet overlapping) subsets of the training data [40].

Reliably measuring the diversity of ensembles is still an unsolved problem, but there are several metrics that are known to correlate with ensemble accuracy. In addition, most of the metrics used are strongly correlated with each other; that is, they appear to measure the same attributes of the ensemble. An important finding is that all diversity measures have a threshold for guaranteed improvement; any ensemble with a diversity greater than this threshold is guaranteed to improve over the single best model it contains. In the general

case, the *Q-Statistic* is recommended as the most widely applicable and intuitive metric [30].

Regardless of how well models in an ensemble are chosen, however, the overall ensemble is useless if it does not combine predictions intelligently. There are two general methods for properly combining predictions. The first, *dynamic classifier selection*, attempts to select the most relevant model for the given sample of data and only uses the output from that model. The second, *classifier fusion*, involves generating a whole new prediction from the predictions of individual models run in parallel. Classifier fusion is the most relevant to this project because it is practically designed to integrate multiple data sources rather than simply getting the most accurate prediction on any given event. The best way to implement classifier fusion depends on the nature of the problem, but there are several standard methods [40]. Nevertheless, care should be taken when choosing and combining models; in one case, the removal of the worst-performing classifier in a five-classifier ensemble actually improved the ensemble results [49]. The choice of fusion technique may also depend on the quality of the models involved. Dynamic classifier selection generally works best when the models in use have been optimized, but ensembles using simple voting techniques did not show noticeable improvement (and occasionally performed worse than their component models) when using optimized models. In general, both algorithm types work best on large or complex datasets, which is what ensembles are most useful for to begin with [49] [40].

2.4 Data Visualization

In attempting to maximize true positives and minimize false positives when monitoring a network for threats, analysts will frequently strategically divide the workload between computer and human actors to better leverage the strengths of each [2]. While machine-based analytics are adept at processing large sets of data and deriving mathematical conclusions about the environment, the human eye far surpasses the performance of computers when discovering and understanding patterns in a visual layout of the data [35]. For this reason, the field of cybersecurity visualization aspires to blend both roles into a single application, one in which large sets of abstract (non-physical) data are sensibly presented on a visual canvas to aid human operators in locating important patterns [2].

The process of converting collection data into a graphical display within a visualization is divided into the following three steps [2]:

1. **Data Transformation:** The process of mapping raw, unprocessed data into more organized data tables with consistent metadata schemas.
2. **Visual Mapping:** The process of creating structures with spatial and graphical properties out of aspects of the data tables.
3. **View Transformation:** The process of developing encompassing views of the visual structures by manipulating graphical parameters (e.g., position, scale, color, shading).

A primary feature of visualization applications actively used in threat detection (compared to applications that passively summarize the results of past threat detections in graphs)

is the tool's interactivity, providing the analyst using the tool with settings to alter the display to better locate or understand visual patterns in the data [2]. Interaction features in visualizations are critical to aiding the analyst in discovering patterns in the data, primarily due to the immense size of many datasets that require analysis [35]. When records on the order of millions or billions are all visually presented to the operator on a single canvas, there is a significant risk of losing important patterns in the clutter of surrounding data [35].

For this reason, many visualizations are outfitted with interactive features for the analyst to manipulate and reduce the onscreen data based on both inherent knowledge about their dataset and any noticeable patterns that appear [2]. By following the principle of "Overview - Filter - Details on Demand," visualizations can leverage interactivity in order to both present the analyst with the entirety of their dataset and help them focus on patterns pertinent to threat detection. For instance, many visualizations support on-demand filtering and clustering according to shared properties, utilizing Gestalt principles to convey meaningful association of the reorganized data to the operator [35]. Other interactive features outside this principle will typically provide general support to the analyst's task, such as saving sessions, creating labels or shortcuts for important data points, or dumping the underlying records surrounding a relevant section of the visualization into external files for further analysis [2].

In the field of network security, visualization tools are particularly useful for conducting analytics due to their numerous methods for visually presenting complex datasets. As lateral movement (by definition) involves the exploration of a malicious actor through various nodes (i.e., computers and servers) of a network, detection processes seeking to conclusively determine if lateral movement has occurred must incorporate multiple dimensions of information into their analysis. In particular, understanding the connectivity of different nodes will allow the operator to determine whether anomalous events occurring at different parts of the network are related, and the times of these events can convey a possible order in which the nodes were compromised [31]. To effectively negotiate both the time and location of various network events and judge the probability that an anomalous event indicates lateral movement, visualizations can use various properties of the data as dimensions in visual structures. For instance, a network graph consisting of connected nodes can use line thickness to represent the level of traffic between connected computers and a color gradient for the degree of the anomaly, while a sliding bar can manipulate time to view this activity at various points throughout the dataset [23]. Figure 2.3 provides an example of a visualization in which communicating nodes are connected with lines and clustered node communities are denoted with colors [31].

Lateral Movement Visualizations

To aid in the search for malicious lateral movement across networks, various visualizations have been developed for ingesting network activity records and mapping the events. The type of interaction (e.g., rule-based or manual filtering, anomaly-based or signature-based detection) and presentation style will vary depending on the available information, but lateral movement visualizations will generally allow analysts to sift through a very large collection

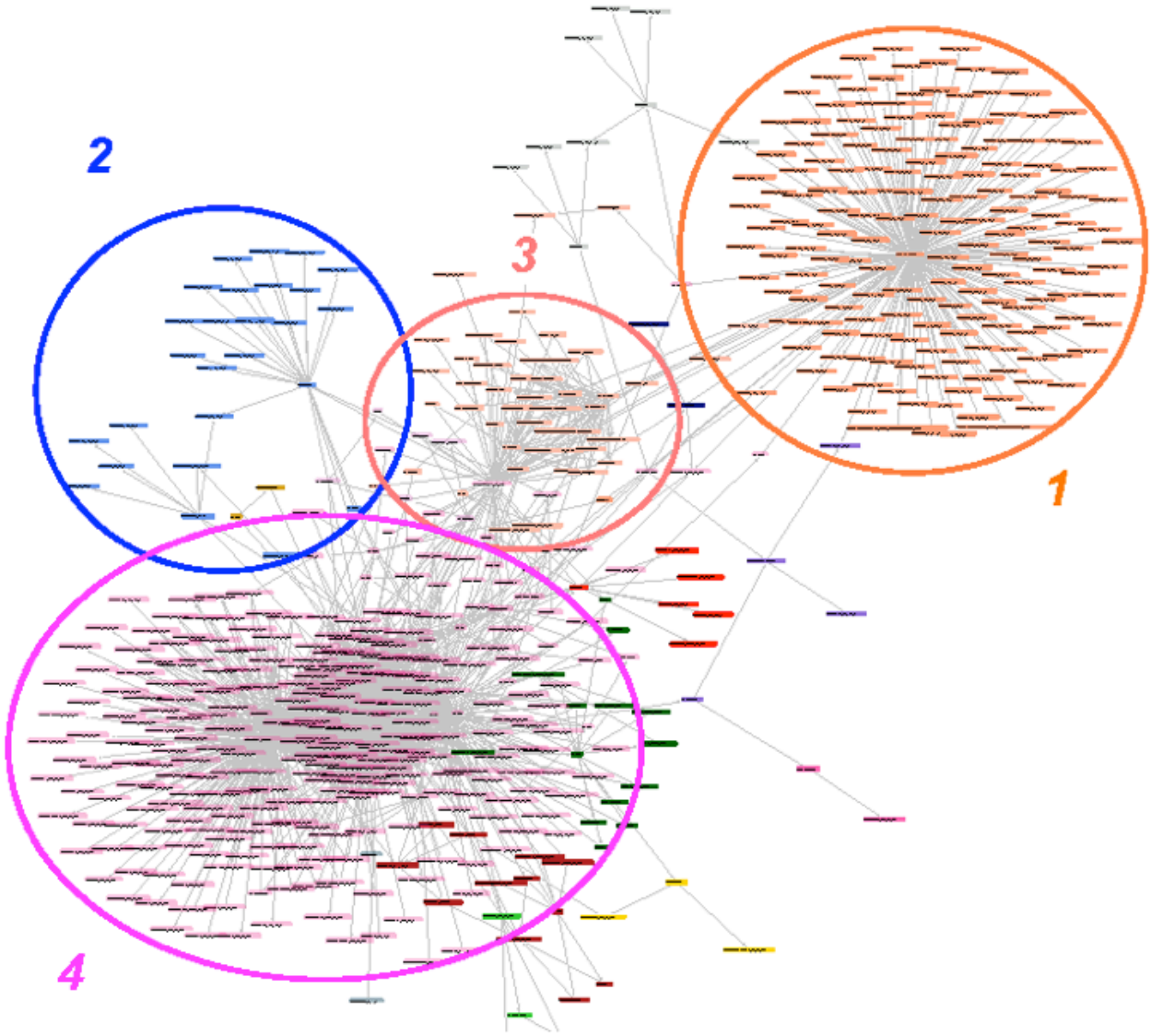


Figure 2.3: Network visualization that uses colors and lines to demonstrate node clustering and connectedness, respectively [31].

of network data, isolate suspicious activities, and follow up on any highlighted records [5]. By working with these visualizations to develop initial suspicions regarding possible lateral movement, analysts can quickly focus on the most important records and subsequently confirm or deny an APT's presence with greater confidence.

One of the most popular lateral movement visualization methods involves the use of graphs, frequently depicting network computers or servers as nodes and their internal communications as edges [36]. Closely mirroring the usual depiction of computer networks as connected graphs, these visualizations provide an intuitive display of the network's layout and - when properly filtered - allow analysts to quickly discover suspicious relationships between users and network resources [34]. Figure 2.4 shows a filtered view of network communications to demonstrate lateral movement between two computers [5].

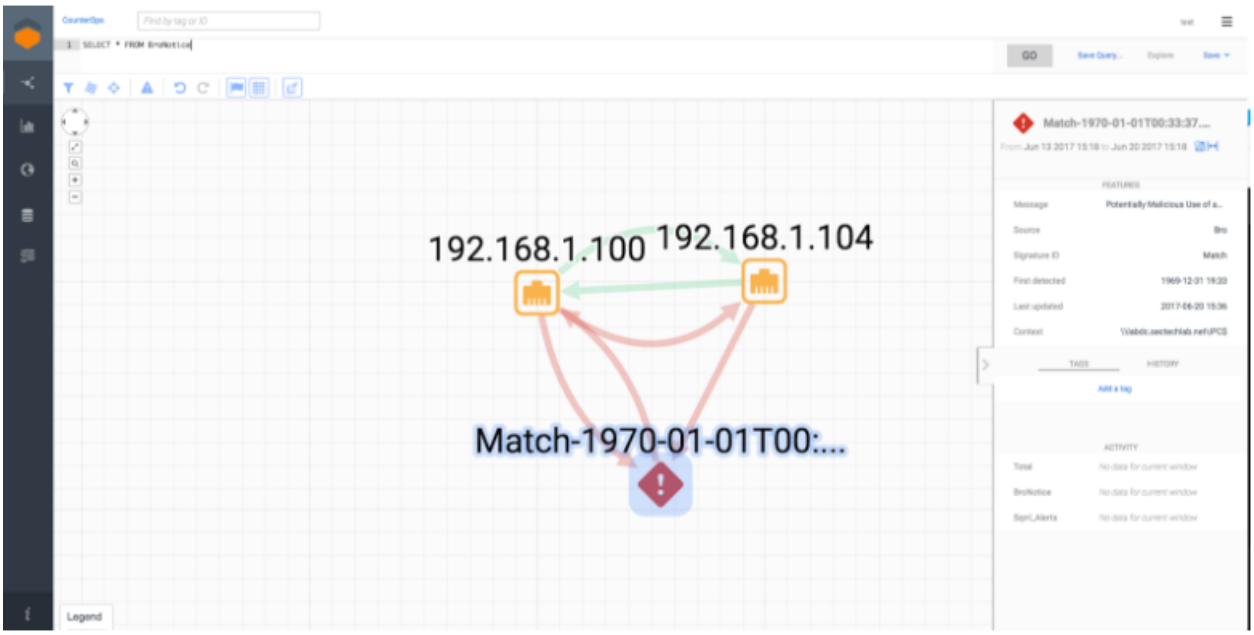


Figure 2.4: Lateral movement network visualization filtered to display the use of PsExec to move maliciously between two computers, identified as 192.168.1.100 and 192.168.1.104 [5].

Furthermore, if IP address data is available to supplement user and computer details in authentication records, graph nodes can be clustered into subnets, with notable communications between nodes of separate subnets clearly visible for analysis [36]. However, when authentication records do not have sufficient information to properly organize the network nodes, this type of visualization can rapidly gain complexity as the number of edges grows, requiring longer analyses to sort the arrangement of nodes into an understandable view [29]. Figures 2.5 and 2.6 depict graphical representations of standard and administrator user communications, demonstrating the noticeable difference in complexity between the two user graphs [29].

Another type of visualization used to depict lateral movement is a two-dimensional matrix, which arranges network communications in a grid according to two individual fields in the network records (e.g., time and IP address, or time and user). Matrices frequently provide

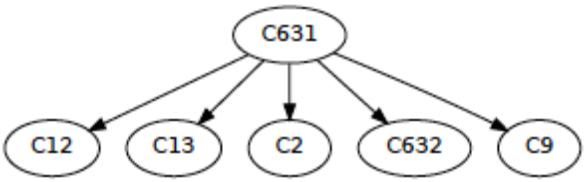


Figure 2.5: Network authentication graph from a typical user *without* administrative access at Los Alamos National Laboratory, over a four-month period [29].

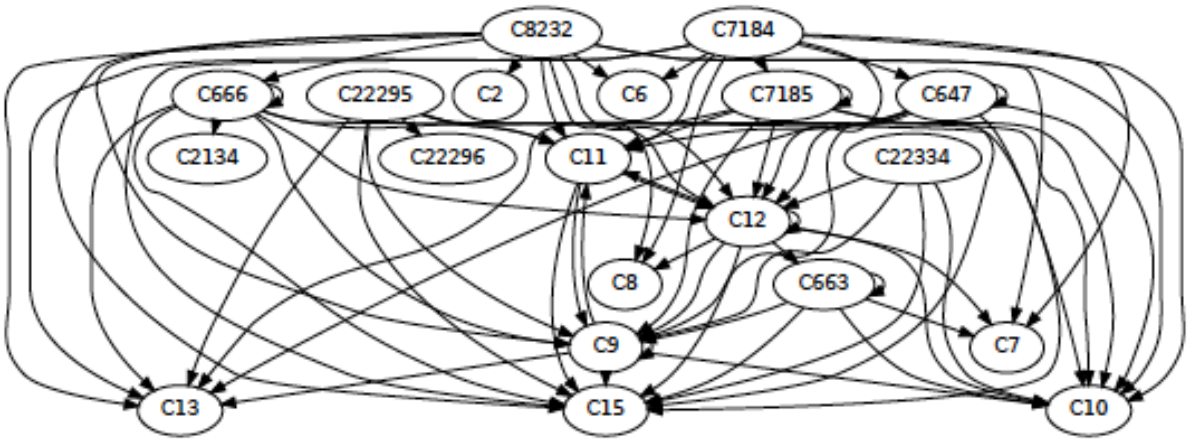


Figure 2.6: Network authentication graph from a typical user *with* administrative access at Los Alamos National Laboratory, over a four-month period [29].

information in the form of pixel maps (specifically outlining patterns of events in sparse grids with large open spaces) or heatmaps (which shade each square to depict the intensity of the event it represents) [5] [18]. These visualizations are useful for their ability to show noticeable patterns and outliers in data when the number of fields in the records are limited, but the number of total records can be quite large; as such, they are particularly well-suited for identifying overarching patterns among the vast number of users and their network activities [5]. Figures 2.7 and 2.8 show an example of a clear pattern that arises from a matrix visualization of user activity profiles: Figure 2.7's depiction of normal network activity shows no regular trend, while Figure 2.8's depiction of records with anomalous activity clearly demonstrates an outlier from normal behavior in the form of an intense column of access in a short moment of time [18]. Once the analyst has identified all of the profiles demonstrating these anomalies, they can return to the records used to build the visualization and investigate further activities performed by the suspicious users (such as usage of PsExec as identified by network sniffers).

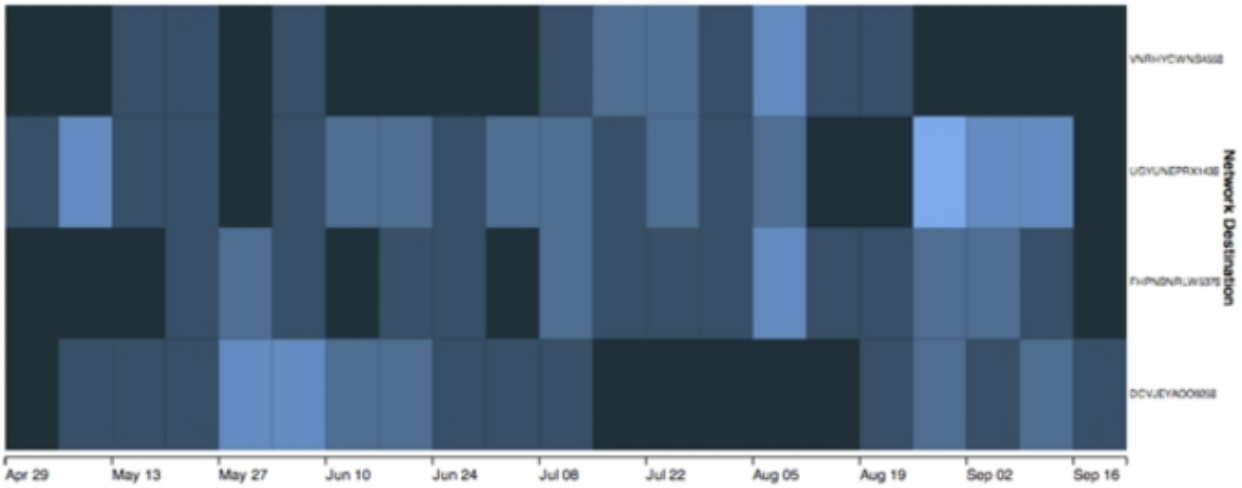


Figure 2.7: User profile heatmap with normal activity. The X-axis is time, while the Y-axis is resources accessed by the isolated user [18].

Both graph and matrix visualizations provide useful information to the analyst regarding the state of the network and potential trends of anomalous activity that may require further investigation. By filtering the initial appearance of the visualization, either automatically through coded rules or manually via clustering or changeable thresholds, the analyst can quickly begin compiling a list of network computers, users, IP addresses, or other identifying features to continue investigating [5]. A particularly effective visualization for guiding the analyst in their initial triage of lateral movement will provide all of these features in an intuitive manner. Such a visualization will generally initially provide the analyst with a comprehensive view of the network records, then filter the appearance to remove clearly benign or irrelevant events, locate noticeable patterns in the events that remain, and - finally - extract raw data from particularly suspicious areas to conduct forensics.

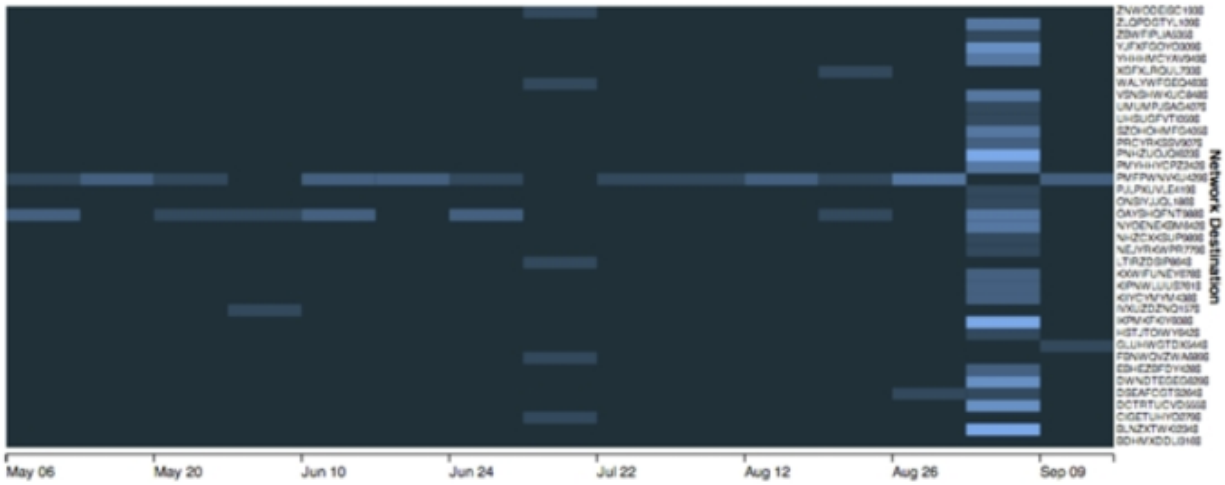


Figure 2.8: User profile heatmap with notable anomalous activity between August 19 and September 9. The X-axis is time, while the Y-axis is resources accessed by the isolated user [18].

2.5 Cybersecurity Datasets

Access to data that is both realistic and abundant is essential for the development and evaluation of data analysis and visualization tools. Unfortunately, data sources that capture lateral movement, or APTs in general, are rare. There are three main reasons for this:

1. Cybersecurity data sources are sensitive; an organization that releases data containing traces of an APT may be announcing security vulnerabilities and putting itself at risk.
2. The instruments (either software or hardware) needed to adequately document events within a network are expensive; this makes it relatively rare for organizations to have sufficient instrumentation.
3. APTs are uncommon when compared to other cybersecurity threats; thus, many data sources simply do not contain any APT activity.

This MQP addressed these issues by considering publicly available datasets that contain redteam activity. Because such datasets are public, we did not need to collect our own data or risk releasing sensitive information. Additionally, datasets that contain redteam data will likely exhibit patterns similar to APTs. With this in mind, this MQP primarily considered the Comprehensive Multi-Source Cyber Security Events dataset published by Los Alamos National Laboratory. However, in the case that this data source proved to be inadequate for the needs of this MQP, we planned to consider datasets including the KDD-99, Mid-Atlantic Collegiate Cyber Defense Competition, CTU-13, or those listed at [43] and [6].

Comprehensive Multi-Source Cyber Security Events

Los Alamos National Laboratory supports a publicly available dataset referred to as the *Comprehensive Multi-Source Cyber Security Events*. The events in this data were collected from the laboratory's internal network across a 58-day period. This dataset – which we will refer to as the Los Alamos dataset – contains five collections [28]:

1. **Authentication Events:** Microsoft Windows based authentication events collected from individual computers, servers, and Active Directory.
2. **Process Events:** Microsoft Windows based stop and start events of processes running on individual computers and servers.
3. **Network Flow Events:** Network flow events collected from central routers within the network.
4. **DNS Events:** Lookup events collected from central DNS servers in the network.
5. **Redteam Events:** Specific events from the authentication data that are indicative of known redteam activity (controlled attacks simulated by Los Alamos personnel to expose network vulnerabilities).

User-Computer Authentication Associations in Time

In addition to the *Comprehensive Multi-Source Cyber Security Events*, Los Alamos National Laboratory supports the *User-Computer Authentication Associations in Time* dataset. This dataset represents the successful authentication events within the laboratory's internal network across a period of nine months [20].

KDD-99

The International Knowledge Discovery and Data Mining Tools Competition sponsors the *KDD-99* dataset, which was used in the 1999 competition. This dataset contains connection events that are labeled as either *malicious* or *benign* [32].

Mid-Atlantic Collegiate Cyber Defense Competition 2012

The *Mid-Atlantic Collegiate Cyber Defense Competition* is an annual cybersecurity competition for college students. These students try to defend a network that is under attack from a redteam. This dataset represents the packet captures (via Wireshark) from 17 unique teams [3].

This dataset has been publicly analyzed by BRO, a popular intrusion detection system, and was found to contain events representing connections, DHCP, DNS, files, FTP, HTTP, notices, signatures, SMTP, SSH, SSL, tunnels, and miscellaneous activity [43].

CTU-13

The *CTU-13* dataset is a network traffic dataset consisting of 13 different malware captures. The events represent normal background traffic and malicious botnet traffic. The data is available as both network flows and packet captures [17].

2.6 Related Works

Although our project built on several established technologies, there were few other projects that combined them in quite the same way. We found two tools that related to our intended project; the APT-Hunter [44] and the Suspiciousness Cascading Graph Detection Method (SCGDM) [25].

APT-Hunter

The APT-Hunter is a tool designed to help analysts detect suspicious authentication activity, which is strongly associated with lateral movement. The tool is designed to leverage the domain knowledge of analysts and aid their intuition with a visualization and a signature-based alert system. The visualization is a simple node-link diagram of the network being monitored, with nodes linked via authentication events and grouped by location. The analyst can filter events, view metadata as layers, and - most importantly - add rules to classify events as either benign or malicious activity. The addition of a rule can be accomplished with a simple interface. The language created for the APT-Hunter allows the analyst to define rules that can be general or specific and optionally focus on allowable times, maximum login failures, and frequency of logins as further restrictions. Although the APT-Hunter requires significant domain knowledge, it effectively leverages the insight of the analyst while minimizing their exposure to false alerts [44].

The benefit of this type of tool is that it is transparent to the analyst and improves over time as they learn more about the network and create more rules. The drawback, however, is that attackers who know the rules in use can hide their activity among benign behavior [44]. The APT-hunter is related to our project because it provides a full backend and visualization, but allows the analyst to decide whether a given pattern of alerts is an indication of lateral movement. It gives more control to the analyst than our framework does, which could make it more effective in the right hands, but it is also less user-friendly and requires a greater investment before it can begin generating useful alerts.

Suspiciousness Cascading Graph Detection Method

The Suspiciousness Cascading Graph Detection Method (SCGDM) is an anomaly-based approach designed specifically to detect lateral movement attacks. The SCGDM tool tracks patterns in suspicious activity across hosts that have communicated with each other in order

to build a *suspiciousness cascading graph*. Nodes in the directed graph represent individual hosts, and edges between hosts are created when one host appears to have compromised the other. This *intrusion causality* is defined as a pattern in which a normal host becomes suspicious shortly after communicating with a suspicious host. Data from multiple sources is used to detect suspicious activity and rate the suspicion level of each host at any given time. A host that becomes suspicious shortly after communicating with a suspicious host is added to the graph containing the first suspicious host in the relationship. An alert is raised when the *suspiciousness cascading graph* this generates exceeds a certain size, which varies depending on the network being monitored. However, since normal network activity is unlikely to cause this type of pattern, the threshold can be low [25].

The SCGDM tool includes a visualization that clarifies the pattern of lateral movement and can help analysts determine the status of an attack. However, its main focus is the detection method itself rather than the visualization; in this point, it differs significantly from our project. In addition, unlike the signature-based method used as a benchmark, the tool could detect stealthy techniques such as the misuse of trusted software [25]. Analyst fatigue should be minimized because the false positive rates and false negative rates are low. Although SCGDM was not tested in a large-scale environment, it is a promising tool to use as a reference. Due to constraints on time and resources, we did not implement an analytic that acts like the SCGDM; however, we used similar ideas as inspiration for our project. Most notably, the "cascading" patterns of suspicious behavior revealed in the SCGDM used the same basic justification as our visualization.

Chapter 3

Methodology

3.1 Formative Study

We devoted the initial two weeks at the Lincoln Lab to properly scoping our project into an attainable but significant contribution to Advanced Persistent Threat (APT) detection. To begin, we completed a literature review on current papers to discover the extent of previous work and research in the subject. While a research librarian at Lincoln Lab's Library and Archives center provided us with a large corpus of scholarly articles and research papers through searches using the terms "lateral movement," "cybersecurity," "APTs," "SIEM," and "privilege escalation," we personally found additional sources through specialized search tools such as Google Scholar. Appendix B lists a subset of the search terms we used.

After learning about projects such as the APT-Hunter [44] and the Suspiciousness Cascading Graph Detection Method (SCGDM) [25] through our literature review, we decided to continue research into lateral movement and its detection process by leveraging ensemble learning and data visualization to better analyze and present network data. Following this decision, we planned a proof-of-concept framework capable of reading network records into a database, analyzing them, and then visualizing the data for analyst interpretation.

As we designed our project to primarily aid network analysts in their detection of APTs and lateral movement, a fundamental aspect of the project's conception involved speaking with analysts and obtaining their insights regarding our decisions and designs [50]. As Hypothesis Two specifies that visualizations mapping time and network host location will aid analysts in their efforts to discover lateral movement, a thorough evaluation of our final product through analyst use and feedback was essential to testing the claim. For these reasons, a thorough test of our Hypothesis Two required a final review by cybersecurity analysts at the Lincoln Lab.

Our final evaluation was centered around interviews with analysts, but we understood the importance of meeting with the analysts beforehand to discuss our project at each stage. In order to improve the final framework design's usefulness to the analysts, we required input both before and during the design process. We obtained this input through iterative

interviews with Lincoln Lab analysts, which we conducted as often as possible (unfortunately limited due to the fast-paced and busy nature of the cybersecurity analyst's job). At each interview, we reviewed our progress and design decisions, particularly in regards to the end-user visualization. This process enabled us to enter the evaluation phase with a proof-of-concept framework designed to satisfy the needs of the particular analysts who would test it.

To this end, we separated our interactions with analysts into three major categories, roughly corresponding to periods before, during, and following development:

1. **Requirements:** Obtaining initial guidelines from analysts or related Lincoln Lab staff regarding an overview of lateral movement and APTs, their significance in cybersecurity, and major intrusion detection methods. This also involved feedback for our proposal presentation and the framework design we outlined therein, which we used to define a relevant and realistic project scope. Appendix A outlines the default list of questions used during Requirement interviews. However, because these were semi-structured interviews involving employees with different specialties in the Lincoln Lab, the exact set of questions asked differed significantly in some cases.

2. **Iterative Review:** Showcasing the latest state of the visualization to obtain feedback. In particular, we used these reviews to revise the visualization in order to maximize the chance that analysts using it could confirm or deny the presence of lateral movement with reasonable confidence. The analysts could also suggest features at this stage; depending on the status of the project, we would either integrate these or include them in our suggestions for future work.

3. **Evaluation:** Proving or disproving the framework's effectiveness in a mock field test. While running the visualization on an experimental dataset with redteam and benign users, the analyst would attempt to determine which users participated in lateral movement. More details regarding the evaluation step can be found in Section 3.4: Visualization and Evaluation of Hypothesis Two.

In our attempts to interview analysts during all three interaction periods, we sought the opinions of personnel in both the Information Services Department (ISD) and the Lincoln Research Network Operations Center (LRNOC) at the Lincoln Lab. ISD provides Information Technology (IT) services to the entire laboratory, including IT security pertaining to the safety of critical information stored on employee computers. ISD's direct efforts in resolving any active cybersecurity threats to the Lincoln Lab provided a relevant background for questions related to intrusion detection and threat resolution, along with access to analysts who could help evaluate our project [1]. Conversely, LRNOC employs operators dedicated to actively monitoring the Lincoln Lab network data for any evidence of cybersecurity threats and bolstering the network's defense against existing threats [41]. We primarily relied upon LRNOC personnel to provide information regarding the analyst's approach to cybersecurity visualizations, while we interviewed several ISD analysts who were willing to participate in our full evaluation process. The specific information obtained from each interview can be found in Section 5.1: Interviews with cybersecurity analysts.

3.2 Framework Overview

This project required us to develop software infrastructure, hereafter referred to as the *framework*. The framework served two purposes:

1. Provide a suitable analysis environment needed to experimentally validate our hypotheses based on actual data known to contain redteam events.
2. Provide a proof-of-concept architecture to use for experimentally implementing visualization strategies and anomaly detection techniques.

The backend of the framework was primarily used to evaluate Hypothesis One, and the frontend of the framework was primarily used to evaluate Hypothesis Two. Our evaluation methods for the individual hypotheses are described in the next two sections. Further details on the framework components and their implementations – as well as the database structure – are provided in Chapter 4: Framework.

Evaluation of Framework

To meet the needs of the experiments we would conduct to test our two hypotheses, we needed to create a framework that would support a sufficient set of use cases. In the backend, the framework's user may:

- Insert any number of cybersecurity datasets, assuming that the following conditions are met:
  - The machine running the framework has enough disk space to store the dataset
  - The dataset is formatted as a CSV file
  - A configuration file is provided
  - The dataset is labeled with ground truth
  - Each even in the dataset has an integer-based timestamp with a relative start time
  - The dataset has one or more location fields
- Insert any number of anomaly detectors as plugins, assuming that the following conditions are met:
  - The machine running the framework has enough disk space to store the plugin
  - The plugin is executable
  - The plugin can interface with the framework's database
  - The plugin is aware of the framework's database schema
  - A configuration file is provided

  - The plugin generates predictions for suspicion levels of individual events on a scale from *benign* (0) to *malicious* (1)

- Select any of the inserted datasets to be used throughout the analysis
- Select any of the inserted anomaly detectors to be used to make predictions on the events in the dataset
- Configure the ratios of the dataset that will be partitioned into training data and testing data
- Create an ensemble out of the anomaly detectors to generate a single prediction by combining detector predictions
- View metrics regarding the performance of the anomaly detectors and ensemble that was used in the analysis
- Select which location fields will be used as dimensions to generate aggregates

In the visualization, the user may:

- Select the location fields that they wish to view as the X and Y axes, assuming that the corresponding aggregates table has already been generated
- Hover over a cell to view its row and column values, along with the volume of activity associated with this pair in the dataset
- Sort rows or columns to view them in descending order of activity, or rearrange particular rows or columns to move them to the map's upper-left corner
- Filter rows or columns based on the analyst's desired number of unique blocks in a row/column, thereby only mapping the events with a specified activity level
- Isolate a specific row for a detailed timeplot of events corresponding to the specified location
- Shade the isolated timeplot blocks in a gradient according to a specified detector or ensemble's suspicion levels of the associated events
- Dump the isolated timeplot's raw data into an external log file to share with other analysts or compare against other network data

If the framework incorporates all of these use cases, it would successfully serve as our proof of concept tool. The specific Engines used in our framework are described below.

Framework Components

As shown in Figure 3.1, the framework developed in this MQP contains 13 logical units.

1. **Database:** Stores the data for by the components of the framework

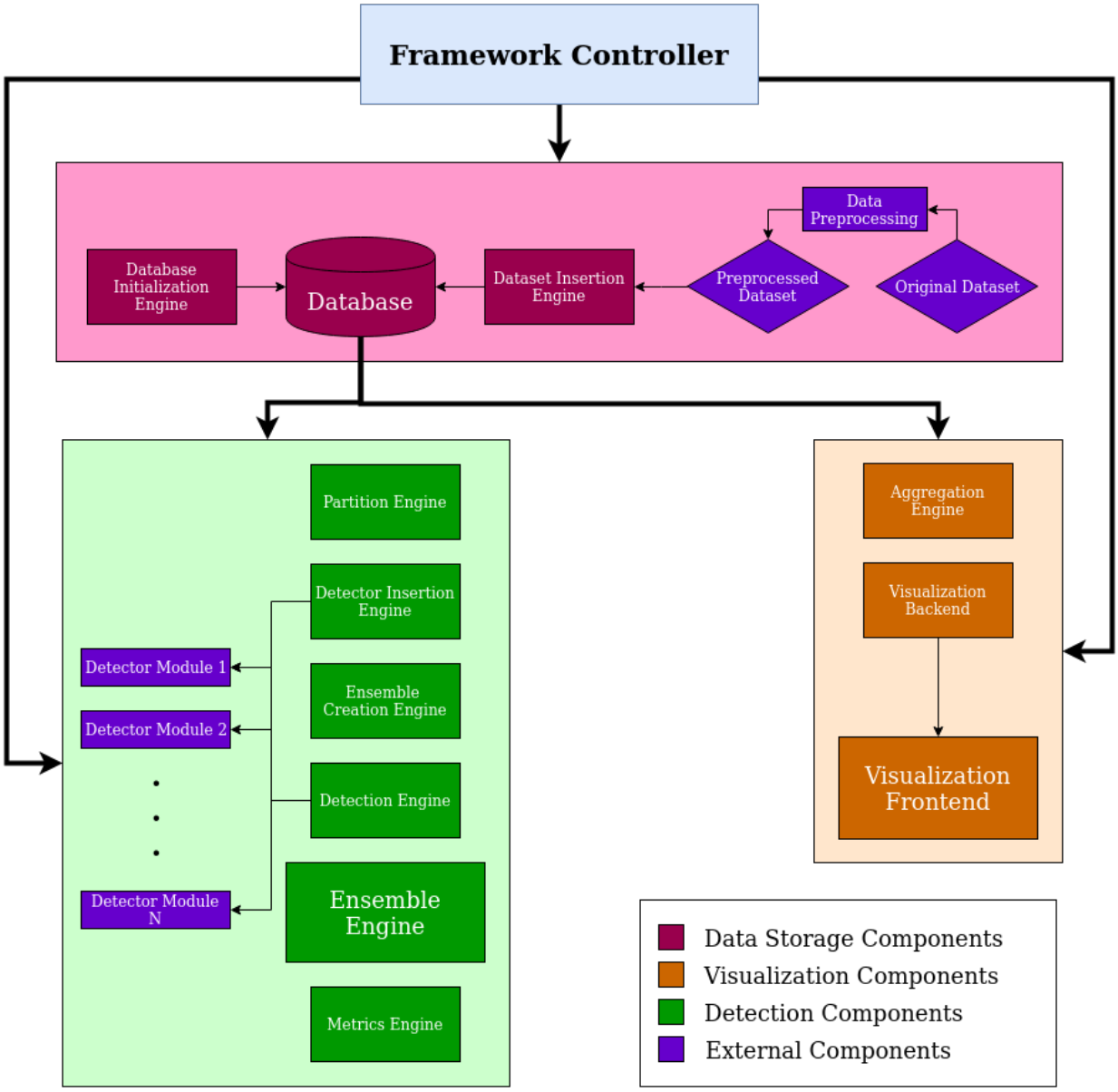


Figure 3.1: The individual components of the framework.

2. **Database Initialization Engine:** Facilitates the initialization of the database schema within the framework
3. **Dataset Insertion Engine:** Facilitates the insertion of datasets into the framework
4. **Aggregation Engine:** Facilitates the aggregation of events based on common values of shared fields
5. **Partition Engine:** Facilitates the partitioning of events within the database into a training set and a testing set
6. **Detector Insertion Engine:** Facilitates the insertion of anomaly detector plugins into the framework
7. **Detector Engine:** Facilitates the detection of malicious events using the individual anomaly detectors
8. **Ensemble Creation Engine:** Facilitates the creation of ensembles of the various anomaly detectors
9. **Ensemble Engine:** Facilitates the detection of malicious events using the ensembles of anomaly detectors
10. **Metrics Engine:** Facilitates the calculation of various metrics regarding anomaly detectors and ensembles
11. **Visualization Backend:** Facilitates the fetching and preprocessing of data needed by the Visualization Frontend
12. **Visualization Frontend:** Visually presents the data received from the Visualization Backend as a JavaScript web application
13. **Framework Controller:** Coordinates the flow of control between the components of the framework

3.3 Analytics and Evaluation of Hypothesis One

The creation of the ensemble portion of the framework required three major design decisions: how we would handle the data, how we would combine detectors to maximize their effectiveness in an ensemble, and how we would combine individual detectors' predictions into one prediction. We implemented this part of the framework as described in in Chapter 4: Framework and tested it as described below.

Major Decisions

There are several well-known algorithms to combine predictions, but the most widely applicable (when the accuracy of each detector can be reliably calculated) is the "weighted

average" method. This method uses the mean of all detector results, with each detector's prediction weighted based on its accuracy.

We calculated the accuracy of the detectors using the AUROCC metric. We did this be generating a ROC curve consisting of 10 points between 0 and 0.9 (in increments of 0.1). The value at each point represented the false positive rate versus the true positive rate, using that point as the threshold between a positive and negative prediction. Anything greater than or equal to the threshold was taken to be 1, while anything below it was 0. Because the resulting ROC curve consisted of a discrete set of points, we could use simple trapezoidal integration to calculate the AUROCC from it. A more advanced method might have given us more precise results, but would not have been worth the time it would take to implement.

We also calculated the diversity of the ensemble to check a key assumption of ensemble learning. If the detectors were not diverse, then the ensemble would not perform significantly better than the detectors in it; in fact, it could potentially perform worse than any of its component detectors. We used the Q-Statistic to measure diversity because of its versatility; Polikar suggested it as the best diversity metric for general-purpose use [40]. We implemented the Q-Statistic calculation by using the values from the pairwise diversity metric, which we generated by comparing the predictions of each detector against ground truth. The pairwise diversity values are as shown in Table 3.1 [40].

	X is correct	X is incorrect
Y is correct	a	b
Y is incorrect	c	d

Table 3.1: The values used to calculate all pairwise diversity metrics between detector predictions, where X and Y are any two detectors that have been trained against the same dataset.

The value of a is the number of events both X and Y classified correctly, d is the number of events both X and Y classified incorrectly, and b and c are the number of events only X or only Y (respectively) classified incorrectly. The pairwise Q-Statistic is defined as:

$$Q_{(X,Y)} = \frac{ad - bc}{ad + bc}$$

High absolute values of the Q-Statistic (which is bounded between -1 and 1) indicate low diversity, while maximum diversity is indicated by a value of 0, or no correlation between detector predictions. We calculated the overall ensemble diversity by averaging the pairwise diversities between detectors [40]. Ideally, the ensemble's Q-Statistic value would be minimized while its accuracy on the test set would be maximized.

Testing Hypothesis One

We tested Hypothesis One with the following experiment. For the sake of this experiment, Hypothesis One states the following: "An anomaly detector ensemble that combines the

predictions from a sufficiently diverse set of component anomaly detectors will detect malicious events with an AUROCC that is greater than or equal to the average AUROCC of its component detectors."

Experiment

1. Preprocess the Los Alamos authentication dataset to add ground truth labels, remove duplicate and incomplete events, and filter the dataset such that it is a manageable size yet still contains a sufficient number of redteam events for a machine learning model to train on. Insert this dataset into the framework using the **Dataset Insertion Engine**.
2. Develop a Random Forest detector, *Detector A*, that uses all fields in the authentication data as features. Insert Detector A into the framework using the **Detector Insertion Engine**.
3. Develop an Imperfect Oracle detector, *Detector B*, that always generates predictions with 80% accuracy. Insert Detector B into the framework using the **Detector Insertion Engine**.
4. Create an ensemble of *Detector A* and *Detector B* and add it to the framework using the Ensemble Creation Engine.
5. Partition the events into 70% training data and 30% testing data using the **Partition Engine**. We chose these values to ensure that both the training and testing sets would both contain multiple redteam events.
6. Use the **Detection Engine** to train Detectors A and B on the training data.
7. Use the **Metrics Engine** to calculate the AUROCC for Detectors A and B on the training data. The **Ensemble Engine** uses these values to determine the weight of each detector when combining their predictions.
8. Use the **Detection Engine** to make predictions on the testing data with Detectors A and B.
9. Use the **Ensemble Engine** to make predictions on the testing data based on the predictions made by Detectors A and B and their performance on the training data.
10. Use the **Metrics Engine** to calculate the AUROCC for Detector A, Detector B, and the ensemble.
11. Use the **Metrics Engine** to calculate the diversity of the ensemble.
12. Repeat steps 5 through 11 a total of ten times to generate data for ten trials.
13. Compute the average AUROCC across all trials for Detector A, Detector B, and the ensemble.
14. Compute the average diversity across all trials for the ensemble.

We implemented the "Imperfect Oracle" to replace the Neural Network (which we had in our original plan) because of the low AUROCC of the Neural Network and the short timeline of our project. Because we were only comparing the AUROCCs of the detectors and the

ensemble itself, we decided that an actual machine learning model was not necessary to complete this test.

If the absolute value of the average diversity of the ensemble was greater than 0.75, then we would deem the experiment invalid and repeat it with modifications to Detectors A and B until the diversity was less than or equal to 0.75. We chose a value of 0.75 because it appeared to be the greatest value that would safely guarantee an significant accuracy boost, based on the data discussed in a paper on diversity metrics [9]. If the average AUROCC of the ensemble was greater than the average AUROCC of Detector A and Detector B over all trials, then Hypothesis One would be supported. Otherwise, if we could not achieve meaningful results within 10 repetitions of the experiment, Hypothesis One would not have support.

3.4 Visualization and Evaluation of Hypothesis Two

After the framework has successfully processed and analyzed the dataset provided, it can generate a visualization using the times, locations, and suspicion levels of each event. Hypothesis Two states that using a visualization that maps both the time and location of suspicious events will allow analysts to understand whether lateral movement has occurred in their selected timeframe. Following this, analyst interactions with the visualization, including abilities to both zoom out into a broader summary of the data and zoom into specific details about suspicious areas, will form the culmination of the framework's tasks and the primary source of information either proving or disproving this hypothesis.

Visualization Frontend

The visualization is the final component of the framework and the point at which analysts usually interact with the data they have inserted into the database. Information about user actions and the respective data that must be displayed will be relayed between the two database components and the visualization through an API. By viewing the events processed by the backend events presented by time and location, the analyst would ideally observe either distinct patterns potentially representative of lateral movement, or a complete lack of such patterns if no lateral movement exists. If the analyst discovers a pattern, they could conduct forensics on the information in question by viewing the original data records corresponding to the important times and locations in the pattern.

Choosing and fine-tuning a visualization design that would effectively display time and location groupings to the analyst and produce observable patterns in the event of lateral movement was one of the most critical development aspects of the project; even if analysts could theoretically detect lateral movement with a time-location visualization, if our proof-of-concept proved difficult for analysts to use, this would produce inconclusive results for Hypothesis Two. As such, we coordinated with the analysts we interviewed and related Lincoln Lab personnel to brainstorm and refine the visualization layout (which involved deciding

which data dimensions to group together and the overall arrangement of these groupings) to produce a usable GUI. In this way, we ensured that our evaluation would test whether lateral movement could be conclusively tracked through time and location groupings in a good visualization, not whether our visualization was of sufficient quality to perform this test.

Our initial visualization concept was a heatmap that displayed aggregated time/location nodes in a two-dimensional matrix, with time as one dimension and location (user, computer, etc.) as another. For the initial visualization mockup, we used a gradient between green and red to designate the number of suspicious events in this grouping, with the brightest shade of red corresponding to the highest suspicion to draw the analyst's attention. Considering the proof-of-concept nature of our framework, we remained focused on ensuring proper functionality rather than choosing an optimal color scheme. However, following the advice of several of the analysts we interviewed, we updated our color scheme to a gradient between cyan and red to avoid potential issues with color blindness. Several mentioned the idea of additionally changing the size of the square instead of the color, but we ultimately ruled this out due to the complexity of its implementation, both in terms of usability and code design. Figure 3.2 shows a screenshot of a visualization mockup created for this purpose, mapping the redteam authentication events in the Los Alamos dataset with user and hour of occurrence as dimensions.

We proposed that lateral movement, frequently consisting of suspicious actions on a single computer before fanning out into multiple new ones, would become apparent with the presence of diagonal "streaks" of highly suspicious activity shifting between different computers over time. However, after our initial testing with our mockup visualization, we realized that abnormal horizontal lines (representing a single, non-admin, user logging into multiple computers) in a comparison between users and computers was generally more indicative of malicious network activity such as lateral movement.

After completing our initial heatmap visualization, we worked closely with analysts at the Lincoln Lab to brainstorm methods of displaying lateral movement through this type of visualization, along with improving the distinction between visualized datasets with and without lateral movement. We describe the iterative implementation of our visualization in Section 5.1: Interviews with cybersecurity analysts. Our goal in improving the visualization was to primarily ensure its potential for use by analysts in lateral movement detection, pursuing features to better aid the analyst before improving its aesthetic appeal. By maintaining a modular design for the visualization components, we have laid the groundwork for future data scientists to build more elaborate visualizations for threat detection with improved time and resources. Instead of pursuing a similarly ambitious visualization in the time constraints of our project, we elected to ensure the usability and effectiveness of our design for evaluating our hypotheses and performing core tasks.

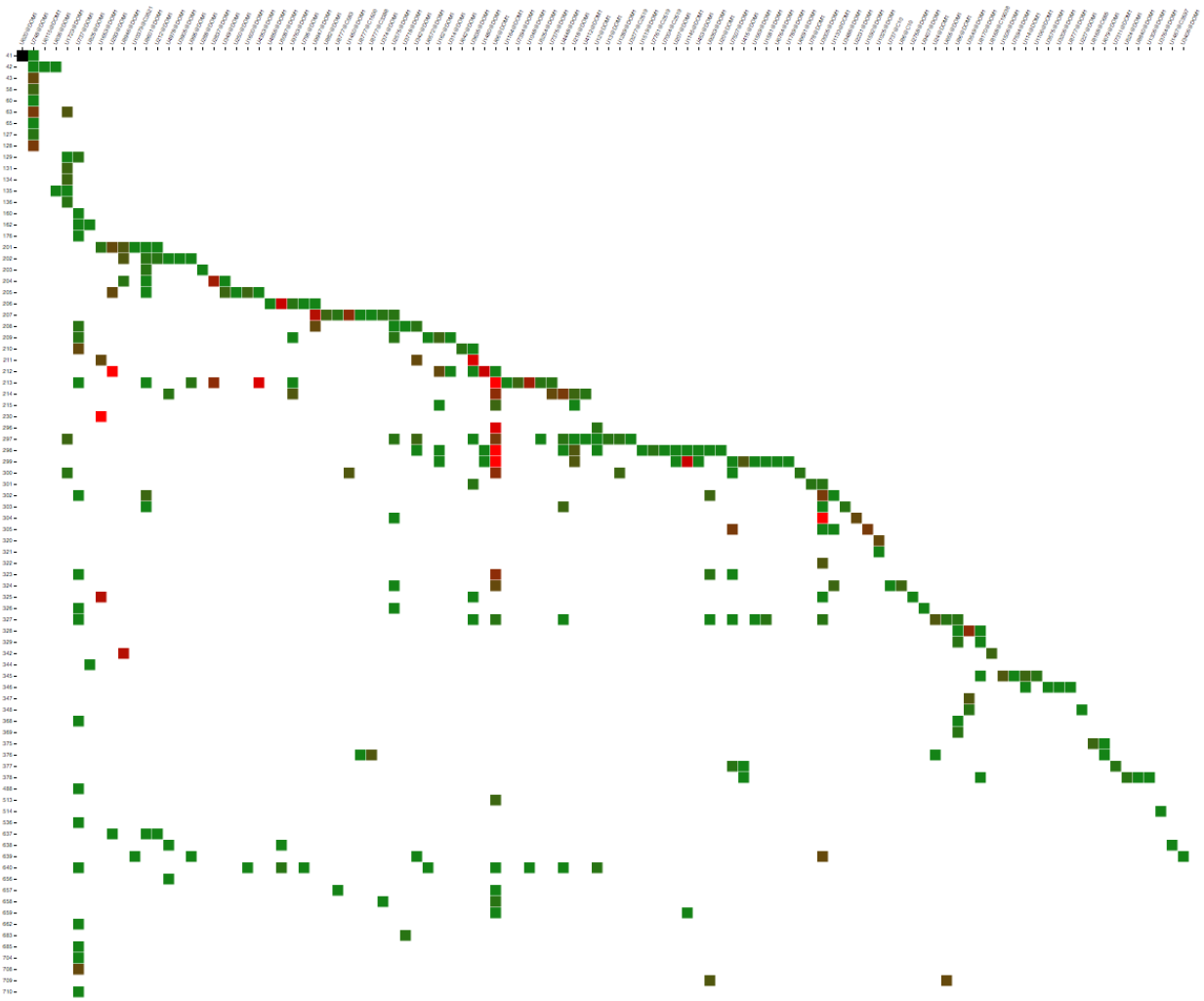


Figure 3.2: A screenshot of an initial heatmap visualization in JavaScript D3. The rows represent hours that contain Los Alamos redteam events, and the columns are the users corresponding to these events. Each square's shading represents how many redteam events occurred in that hour, from 1 (green) to 10 (red). White spaces contain no redteam events.

Testing Hypothesis Two

Hypothesis Two, as presented in the Introduction, was tested with the following experiment. For the sake of this experiment, Hypothesis Two states the following: "A visualization that organizes anomalous events based on each event's time, location, and suspicion level will allow an analyst to determine whether a given pattern of events is caused by lateral movement."

We began with an informal evaluation of our proof-of-concept design with four ISD analysts, two of which had been in our previous interview session (in which they gave feedback on our initial visualization design). This evaluation involved a demonstration in which the analysts could ask questions and give feedback as ideas occurred to them. Although this form of evaluation was mostly qualitative, it was more representative of the type of data we were able to collect given the extremely limited chances to meet with analysts themselves. The results of this interview can be found in Section 5.3: Experimentation for Hypothesis Two. In addition to this informal feedback session, we designed a simple experiment to provide more quantitative results.

Experiment

1. Develop a detector, the *Oracle*, that uses ground truth to detect malicious events. Insert the ground truth detector into the framework using the **Detector Insertion Engine**.
2. Preprocess the Los Alamos authentication data to reduce the size of the dataset, add ground truth labels, and remove any duplicate or incomplete entries. Insert this dataset into the framework using the **Dataset Insertion Engine**.
3. Use the **Detection Engine** to generate predictions on the dataset with the Oracle.
4. Use the **Aggregation Engine** to group events with the same time and location and assign a *suspicion level* to each group.
5. Use the **Visualization Backend** to sort the locations based on order of appearance, then prepare and send the data for the visualization frontend.
6. Use the **Visualization Frontend** to visualize the data.
7. Repeat steps 2 through 6 with a timespan of the Los Alamos dataset known not to contain lateral movement.
8. Present each visualization to at least two cybersecurity analysts. Have them use each visualization to label the corresponding dataset as *contains lateral movement* or *does not contain lateral movement*.

After implementing the framework and attempting to prepare datasets for use in the outlined experiment, however, we realized that our framework's visualization was not powerful enough to manipulate datasets of over 100,000 events without sacrificing usability and responsiveness. To obtain quantitative feedback, we instead designed a new visualization experiment

with a significantly smaller dataset. For more information regarding the limitations of our framework within the scope of this project, see Section 6.2: Discussion of Hypothesis Two.

Experiment (Hypothesis Two Alternate)

1. Preprocess the Los Alamos authentication data to reduce its size, add ground truth labels, and remove any duplicate or incomplete entries.
2. Isolate a dense cluster of redteam events in the dataset, ideally events from approximately four different users conducted within the same 100 seconds.
3. From the chosen time range, randomly select an equal number (i.e., four) of benign users (who have not conducted any redteam activity) and compile all of their events within the same time range, along with the redteam events identified in Step 2, as a new dataset.
4. Insert this dataset into the framework using the **Dataset Insertion Engine**, then use the **Aggregation Engine** to group events with the same locations (i.e., Source User and Destination Computer) in preparation for displaying them in the visualization.
5. Use the **Visualization Backend** to send the aggregated values to the **Visualization Frontend** to display the complete dataset.
6. Individually explain the visualization's use and the reasoning behind it to each subject in the experiment (personnel at the Lincoln Lab with moderate to extensive backgrounds in cybersecurity). Allow them to freely interact with an example dataset containing 10,000 events (none of which are redteam events) for approximately 10 to 15 minutes. Ensure that they are familiar with the user interactions provided (including viewing which computers have received activity from different users and the patterns of authentication each user performed throughout the dataset) and answer any questions they have.
7. Individually present the visualization with the evaluation dataset to each subject, disclosing that the total number of redteam users is between one and seven. Allow the subjects to freely interact with the visualization for approximately 10 to 15 minutes to allow them to get a better understanding of the 100 seconds of activity from eight users. At this point, we would answer any questions that would have the same answers for any arbitrary dataset, but would not answer anything related to the data. This was necessary because our visualization was not very user-friendly or intuitive.
8. Ask each subject to use the information gained from the visualization to predict which of the eight users have been involved in redteam activity.

Considering the sparseness of redteam events in the Los Alamos dataset (which contained an approximate maximum density of only 20 redteam events per 160,000 authentication events), our redesigned experiment allowed us to test the visualization's usefulness at uncovering suspicious network activity in a relatively balanced setting. The reasonable dataset size (237 events) and balance between malicious and benign users (1:1) prevented this experiment from exceeding either the framework's technical limitations or the analyst's ability to locate few threats in a very sparse dataset with limited time.

These factors mitigated many of the technical challenges involved with tracking threats in realistic datasets (which may be almost arbitrarily large), ensuring the experiment would simply test the inherent usefulness of a time/location-based visualization for isolating anomalies in network authentication events. As such, strongly positive results in this experiment would suggest the validity of Hypothesis Two, although the presence of the simplifying factors mentioned above mean the framework would require more robust testing than we could expect to complete given the limitations of our project. However, negative results would provide considerably stronger evidence to disprove Hypothesis Two. If analysts could not use the visualization's time/location groupings to accurately label four out of eight users as malicious in such a small dataset, their likelihood of pinpointing 20 redteam events from over 160,000 events (at best) when handling real-world datasets would be extremely low. Therefore, a low accuracy on this experiment would strongly suggest that visualizations using only time and location values in authentication data are insufficient for detecting lateral movement.

Chapter 4

Framework

4.1 Control Flow

The flow of control throughout the framework is dictated by a shell, which is the terminal-based user interface to the Framework Controller. This shell enables the analyst to control the entire system, while restricting the operations to only those that are safe given the current state of the system. Through interaction with the shell, the analyst can run the various framework components and set up a workspace to perform proper analysis and lateral movement detection.

This approach has a number of advantages, the most prominent of which is the ability to change a later stage's parameters without having to rerun prior stages. Although each stage must run to completion at least once in the outlined sequence before the next stage can be run, the stages can then run at any time by utilizing the relevant data stored inside the database.

The shell supports the following commands:

- *exit*: Leave the shell.
- *help*: Displays the proper usage of the shell and the commands it supports.
  - *help -cmd [command]*: Display the proper usage for the given command.
- *set-credentials [hostname] [username] [password]*: Connect to the MySQL server running on the given hostname using the given username and password.
- *workspace* A set of commands related to workspaces within the framework.
  - *workspace -showall*: List all available workspaces.
  - *workspace -create [name]*: Create a workspace with the given name.
  - *workspace -use [name]*: Use the workspace with the given name.
  - *workspace -delete [name]*: Delete the workspace with the given name.

  - *workspace -clear*: Clear all content from the current workspace.

- *insert*: A set of commands used to insert content into the workspace.
  - *insert -dataset [path]*: Insert the properly configured dataset at the given path into the workspace.
  - *insert -detector [path]*: Insert the properly configured detector plugin at the given path into the workspace.
- *aggregate*: A set of commands used to aggregate events.
  - *aggregate -all*: Aggregate events based on all possible pairs of location fields.
  - *aggregate [field-pair]*: Aggregate events based on the given field pair (specified as Field1_X_Field2).
- *partition [test-proportion]*: Partition the events into a training set and a testing set based on the given proportion of test events.
- *detect*: A set of commands used to generate predictions with the installed detectors.
  - *detect -all*: Detect suspicious events using all anomaly detectors that are currently inserted into the database.
  - *detect [name]*: Detect suspicious events using the anomaly detector with the given name.
- *ensemble*: A set of commands used to combine individual anomaly detectors using an ensemble.
  - *ensemble -create [name]*: Create an ensemble with the given name (specified as Detector1_X_Detetor2).
  - *ensemble -run -all*: Detect suspicious events using all existing ensembles in the workspace.
  - *ensemble -run [name]*: Detect suspicious events using the given ensemble (specified as Detector1_X_Detector2).
- *delete*: A set of commands used to delete content from the workspace.
  - *-dataset [name]*: Delete the dataset with the given EventType name (*-all* for all).
  - *-detector [name]*: Delete the detector with the given name (*-all* to delete all detectors).
  - *-ensemble [name]*: Delete the ensemble with the given name (*-all* for all ensembles) (specified as Detector1_X_Detector2).
  - *-partition*: Delete all partitions.
  - *-aggregate*: Delete the aggregate table with the given name (*-all* for all) (specified as Field1_X_Field2).

- *status*: Display the status of the current workspace.

We implemented the shell using object-oriented design principles and ran it as shown in Algorithm 4.1. We designed an abstract class, *Command*, to define the structure and function of all commands. Each shell Command was created as an object that extended this abstract class and was capable of parsing itself from a line of text, then executing itself in the appropriate manner.

Algorithm 4.1. Algorithm used to run the framework's user shell.

```
1 function SHELL
2     while true do
3         printPrompt
4         command = getLineFromUser()
5         exit = ProcessCommand(command)
6         if exit then return
```

4.2 Database Architecture

We used the MySQL database software throughout this MQP to facilitate the storage of data. Each workspace within the framework was mapped to a new MySQL database. Each of these databases followed the schema shown in Figure 4.1, which was composed of the following 18 tables:

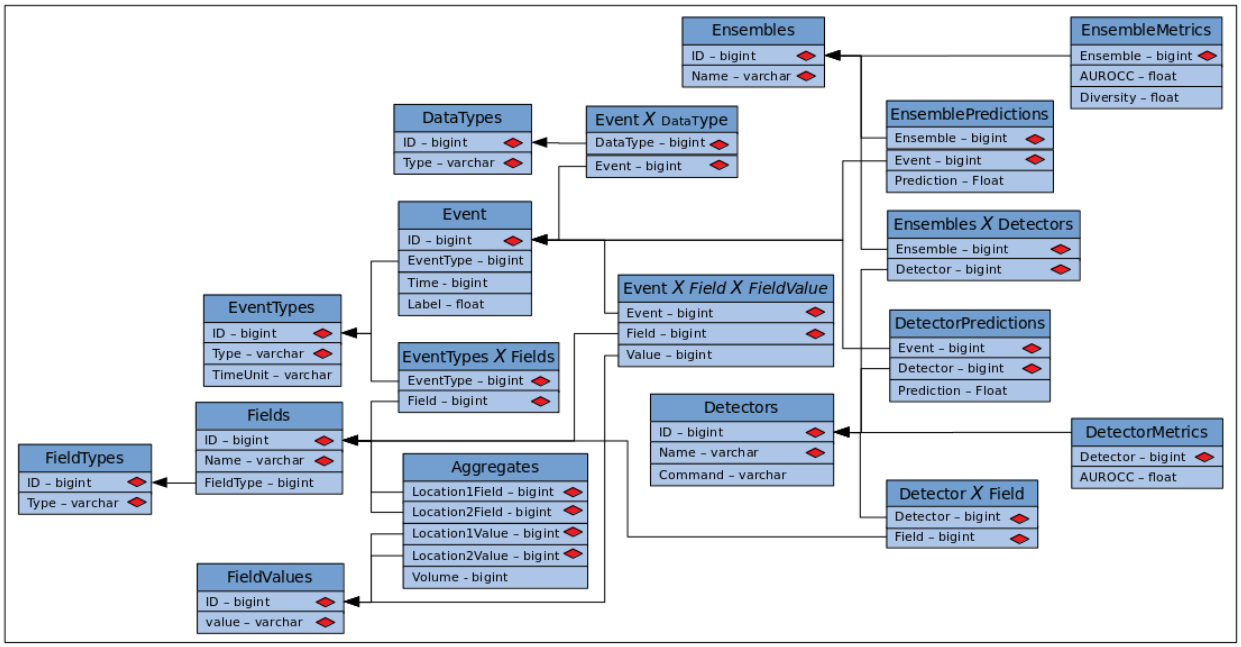


Figure 4.1: This figure shows the tables in the database. The red diamonds represent columns that are part of the primary key. The arrows represent foreign keys.

1. **EventTypes**: EventTypes represent logical groupings of events within a workspace. Within the scope of the Los Alamos dataset, the EventTypes were *Authentication*, *Process*, *Flows*, and *DNS*.
2. **Events**: Events represent a single data point within the dataset.
3. **FieldTypes**: FieldTypes represent logical groupings of fields. This table is initialized with the static values of *Location* and *Normal*.
4. **Fields**: Fields represent an attribute that is shared by multiple events. Within the Los Alamos Authentication data, the FieldTypes included *Source User*, *Destination Computer*, *Source Computer*, and *Destination Computer.*
5. **EventTypes x Fields**: This table links Fields to EventTypes, indicating that the provided EventType has the provided Field.
6. **FieldValues**: FieldValues represent the possible values that can be found for a single field within the data. Within the Los Alamos dataset, the field values included items such as *U66@DOM1*, *C867*, and *Kerberos*.
7. **Events x Fields x FieldValues**: This table links Events, Fields, and FieldValues, indicating that that the provided Event has the provided FieldValue for the provided Field.
8. **Aggregates**: Aggregates represent the volume of Events that occurred for a given pair of Fields and FieldValues.
9. **DataTypes**: DataTypes represent ways to logically partition the data within a workspace. This table is initialized with the static values of *Train* and *Test*.
10. **Events x DataTypes**: This table links Events and DataTypes, indicating that the specified event is a member of the specified DataType partition.
11. **Detectors**: Detectors represent plugins that run externally and label events with a *suspicion level* between 0 (benign) and 1 (malicious).
12. **Detectors x Fields**: This table links Detectors and Fields, indicating that the specified Detector requires the specified Field.
13. **DetectorPredictions**: This table links Detectors and Events, indicating that the specified Detector generated a prediction for the specified Event.
14. **DetectorMetrics**: Metrics represent the performance of the detector, measured as AUROCC.
15. **Ensembles**: Ensembles represent a Detector that uses ensemble learning and the predictions of multiple Detectors to label events as suspicious or benign.
16. **Ensembles x Detectors**: This table links Ensembles and Detectors, indicating that the specified Detector is a component of the specified ensemble.
17. **EnsemblePredictions**: This table links Ensembles and Events, indicating that the specified ensemble generated a prediction for the specified Event.

18. **EnsembleMetrics**: Metrics represent the performance and diversity of the ensemble, measured as AUROCC and diversity, respectively.

These tables are queried and populated by the framework's Engines across a variety of stages, as shown in figure 4.2.

Furthermore, the framework created a user account in each database for the detectors to use. This user account has "SELECT" permission on all tables highlighted in green in figure 4.2f and "INSERT" permission on all tables highlighted in purple in figure 4.2f.

Implications of Database Architecture

We used MySQL as the framework's database backbone because it is a well-supported SQL relational database. This framework required a relational database because event *metadata* was generated across multiple stages. In order to develop a maintainable and clean database, we decided to store this data across multiple tables. Doing so allows a table to be populated in full at a single stage, as opposed to being populated at all stages. With such a design, relational databases achieve better performance than their alternatives. Additionally, because this framework was a proof-of-concept, we required a fully documented and supported database infrastructure to ensure rapid development.

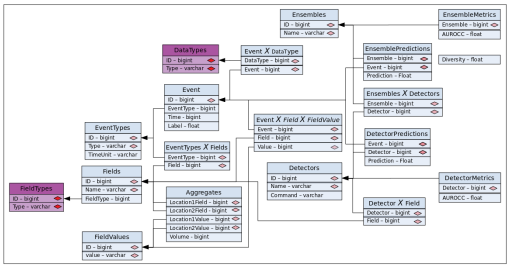
By storing the FieldValues for all Fields and all Events in a single table (Events x Fields x FieldValues), the schema can store any dataset that can be represented as a two dimensional table. Doing so allows the schema to be generalizable and flexible; however, it also introduces limitations. Storing data in this way requires that all FieldValues be stored as strings, which may reduce the number of dimensions they can use. This may result in some loss of information; however, we made this trade-off for the sake of this proof-of-concept framework because it was used to analyze publicly available datasets, which often have already reduced the dimensions of the data for the sake of organization's security and privacy.

The separate MySQL account used by threat detectors allows for a stronger security posture within the framework. It prevents the detectors from exceeding their intended privileges in the database (either by reading from or writing to restricted tables, deleting records, or dropping tables).

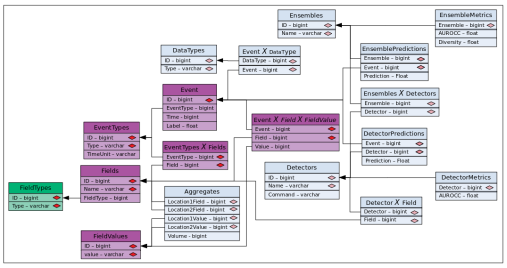
4.3 Framework Components

Database Initialization Engine

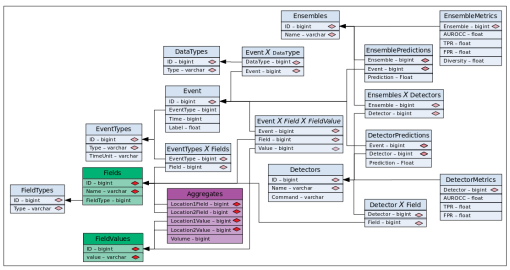
This Engine is run each time a new workspace is created within the framework. It creates a new MySQL database, initializes the database with the schema shown in Figure 4.1, populates the static data tables as shown in in Figure 4.2a, and creates the user account for the anomaly detectors.



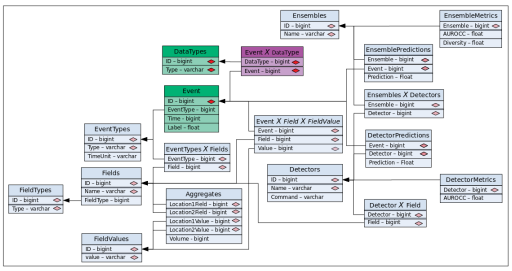
(a) Workspace Instantiation



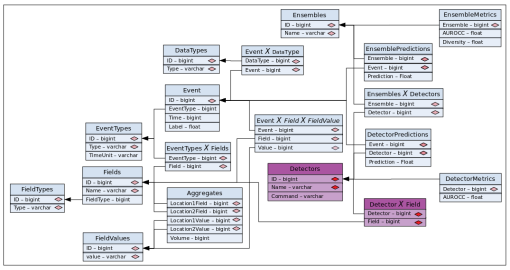
(b) Dataset Insertion



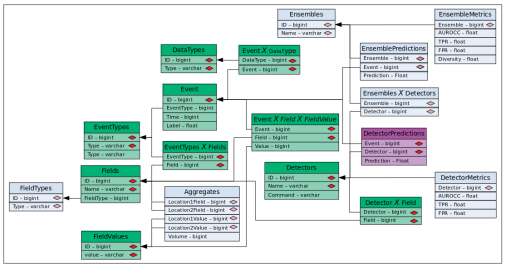
(c) Event Aggregation



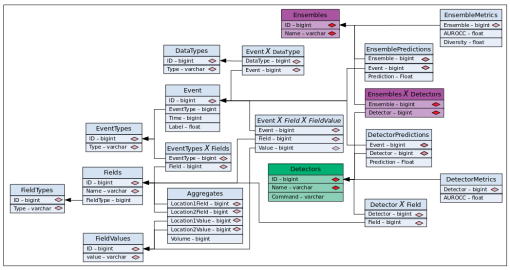
(d) Event Paritition



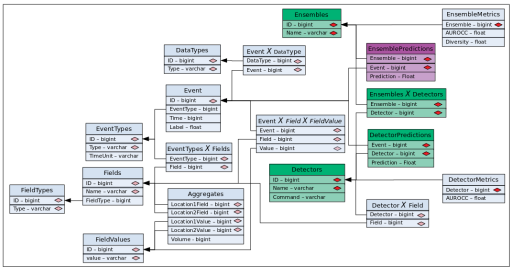
(e) Detector Insertion



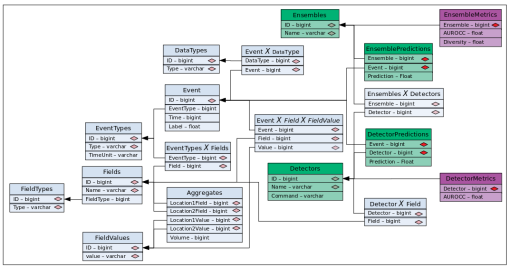
(f) Anomaly Detection



(g) Ensemble Creation



(h) Ensemble Prediction Generation



(i) Metric Calculation

Figure 4.2: This figure shows how the database tables are queried and populated as the various stages of the framework are run. Tables shown in purple are populated at the specified stage, while tables shown in green are queried at the specified stage.

Dataset Insertion Engine

This Engine is run each time raw data is inserted into the database. It supports the insertion of any cybersecurity dataset, provided that the dataset meets the following requirements:

1. Every event in the dataset has a timestamp.
2. Every event in the dataset has a ground truth label with a value between 0 and 1.
3. Every event in the dataset has at least two fields that are designated as *Locations*.
4. Every event type in the dataset is in a separate CSV file.
5. An XML configuration file to designate labels and specifications (time or location) of the dataset's fields, *conf.xml*, is provided with the dataset, and follows the format shown in Figure 4.3.
6. The dataset is formatted as a directory containing the XML configuration file and all required CSV files.

The Dataset Insertion Engine parses the configuration file and builds a model of the dataset to verify that it meets the requirements of the framework. After modeling the dataset, the Engine inserts the events into the database. Initially, this was done one event at a time; however, this resulted in an insertion speed of approximately 9 events per second. This slowness was likely a result of the latency involved with communicating with the MySQL server, as opposed to the actual execution of the query.

To improve the performance of the Dataset Insertion Engine, we implemented a batching approach to insert events in batches of 1,000 rather than individually. Because the database used a relational schema, the implementation of the batching approach was non-trivial. The algorithm maintained a cache of the various FieldValues' Values and their corresponding IDs. This was needed because the FieldValue's Value is known when reading the original data file, but the FieldValue ID is needed as data is inserted into the database. In other words, relational databases allow cells in one table to reference cells in other tables; thus, a cache was maintained to map the references (IDs) to the values (FieldValues). If our algorithm did not use a cache, then the database would need to be queried for each event to obtain the required ID, reducing the performance significantly. However, with the cache-based batch approach, the Dataset Insertion Engine was able to parse and insert more than 1,500 events per second (based on a trial in which the Engine inserted approximately 10,000 events in 6 seconds).

Aggregation Engine

The Aggregation Engine is used to determine the volume of events that occurred at the given timestamp and share the same FieldValue for two of their Fields, provided that each of these Fields is a *Location*. The aggregate of two Fields, Field 1 and Field 2, at time t is the count of events that ocurred at time t and have *FieldValue A* for *Field 1* and *FieldValue B* for

```
<dataset>
   <name> ~~~ </name>
   <eventtype>
      <name> ~~~ </name>
      <file> ~~~ </file>
      <time_unit> ~~~ </time_unit>
      <field>
         <column> ~~~ </column>
         <type>Time</type>
      </field>
      <field>
         <column> ~~~ </column>
         <type>Location</type>
         <name> ~~~ </name>
      </field>
      <field>
         <column> ~~~ </column>
         <type>Location</type>
         <name> ~~~ </name>
      </field>
      <field>
         <column> ~~~ </column>
         <type>Label</type>
      </field>
      ...
   </eventtype>
   ...
</dataset>
```

Figure 4.3: This figure shows the structure of a dataset configuration file. The XML file must begin with a *dataset* tag, which in turn contains a *name* tag and one or more *eventtype* tags. Within the *name* tag, the analyst must provide the name of the dataset. Each *eventtype* tag must contain a *name* tag, a *file* tag, a *time_unit* tag, and a set of *field* tags. The *name* tag must provide the name of the event type, the *file* tag must provide the name of the file within the dataset directory that contains the actual data, and the *time_unit* tag must contain the unit of time used by the timestamps. Each *field* tag is composed of a *column* tag (which references the column within the CSV file that this field is linked to) and a *type* tag (which specifies whether this field is a *Time*, *Location*, *Label*, or *Normal* field). If the field is either a *Location* or *Normal* field, then the analyst must specify a *name* tag, which states the name of this field. In order for an *eventtype* to be valid, it must contain exactly one *Time* field, exactly one *Label* field, and a minimum of two *Location* fields.

Field 2, for all possible pairs of *FieldValue A* and *FieldValue B*. The analyst can request to aggregate along a specific pair of location Fields, or along all possible location Field pairs.

This Engine calculates the aggregates for two Fields using a single SQL query, shown in Algorithm 4.2. This is an efficient approach because it does not require data to leave the database.

Algorithm 4.2. The SQL Query used to aggregate events along two Fields.

```
INSERT INTO Aggregates(Volume, Time, Location1Value, Location2Value,
Location1Field, Location2Field) (
   SELECT A.*, Field1ID, Field2ID
   FROM (
      SELECT COUNT(A.ID), A.Time, A.Field1, A.Field2
      FROM (
         SELECT A.ID, A.Time, A.Field1, B.FieldValue AS Field2
         FROM (
            SELECT A.ID, A.Time, B.FieldValue As Field1
            FROM (
               SELECT ID, Time
               FROM Events
            ) A INNER JOIN EventsXFieldXFieldValues B ON A.ID = B.Event
            WHERE B.Field = Field1ID
         ) A INNER JOIN EventsXFieldsXFieldValues B ON A.ID = B.Event
          WHERE B.Field = Field2ID
      ) A GROUP BY A.Field1, A.Field2, A.Time
   ) A
)
```

This Engine primarily serves to preprocess the data to increase the speed of the Visualization Backend. The visualization requires both a fine-grained time-stamped aggregate and a coarse time-independent aggregate. The fine-grained aggregates are created by aggregating events along their Time and two shared FieldValues. From these values, the coarse aggregates can be computed using the SQL "GROUP BY" clause.

Partition Engine

Each detector in the ensemble requires training and testing, so we developed this Engine to label all events from raw data sources as assisting one of the two detector requirements. In addition, the detectors required a specific method to receive and discriminate between training and testing data. In order for the Ensemble Engine to work properly, each detector must be trained and tested on the same data. Our decision to partition the data in the database into train and test sets was primarily to enforce consistency between detectors. Altering the training and testing sets must not alter the original data, so we dedicated a

separate table in the database to labeling each event in the raw data as training or testing data.

As a parallel to the training-testing nature of our detectors, we designed the Ensemble Engine to function in two stages: training and testing. The training stage dispenses training data to the detectors, combines their results as an ensemble, and calculates the accuracy for each detector and the ensemble as a whole. Each set of training data requires one iteration of this process, and the results are not visualized. Conversely, in the testing stage, the framework behaves normally except for the addition of ground truth, which we would not have in real-world data. Future additions to the framework could add a third phase for real-world data, but this implementation would require specific (and inaccessible) cybersecurity data sources and remains outside the scope of our project.

The Partition Engine is used to split the events into a training and testing set. The analyst can run this Engine with any value between 0 and 1, which specifies the proportion of events that will be designated as testing data. The events are then selected randomly from the raw data until the appropriate numbers have been allocated as either training or testing.

Detector Insertion Engine

The Detector Insertion Engine is used to insert anomaly detector plugins into the framework, provided that they meet the following requirements:

1. The detector can be run from the command line with a single command.
2. The detector accepts the MySQL username, hostname, password, and database name as command line arguments.
3. The detector knows how to properly interact with the MySQL database (i.e., it knows which tables to query and where to insert predictions).
4. A XML configuration file, *conf.xml*, is provided with the detector and follows the format shown in Figure 4.4.

```
<detector>
    <name> ~~~ </name>
    <command> ~~~ </command>
    <requiredfield> ~~~ </requiredfield>
    ...
</detector>
```

Figure 4.4: This figure shows the structure of a detector configuration file. The XML file must begin with a *detector* tag which itself contains a *name* tag, a *command* tag, and any number of *requiredfield* tags. Within the *name* tag, the user must provide a name of the detector. Within the *command* tag, the user must provide the one-line command used to execute the detector. Within each *requiredfield* tag, the user must provide the name of a field that the detector requires in order to make predictions.

Detector Engine

The Detector Engine is used to run the individual anomaly detectors; depending on the input, it can run either a single detector (with the given name) or all detectors. Each detector is executed as an independent process using Java's *exec* feature. Doing so allows for a language agnostic plugin system; as long as the detector an be executed as a shell script, the framework can run it. Because the plugins are language-agnostic, our framework empowers plugins to use the most appropriate language for the task. However, running the detectors in this way introduces security vulnerabilities. A malicious detector could easily exploit the framework to execute arbitrary code. Because this framework was a rapidly-created proof-of-concept – to be used only by our team or under controlled and monitored conditions – we decided to sacrifice the system's security for the sake of flexibility and usability.

For the sake of this MQP, we initially implemented three detectors. The first detector, the Oracle (implemented in Java), simply echoed ground truth labels and served to model an *ideal* detector. The second detector, the Neural Network (also implemented in Java, using the *encog* machine learning library [21]), accepted the Authentication Type, Logon Type, Authentication Orientation, and Success or Failure of each authentication event in the Los Alamos dataset. All possible values were mapped to separate input nodes of the Neural Network so it would use all possible features. This process is common practice when applying discrete features to Neural Networks, which traditionally only expect continuous features. The third detector, the Random Forest (implemented in Python), accepted the Source User, Destination User, Source Computer, and Destination Computer in the Los Alamos dataset. We implemented this model using the *scikit-learn* machine learning library [39] and prepared the data using the *numpy* and *Pandas* libraries for scientific computing and data analysis [24] [33].

Due to the poor quality of the Neural Network detector, we implemented a replacement – referred to as the "Imperfect Oracle" – which added a 20% error rate to the Oracle's predictions. We implemented this random error by assigning an incorrect label to 20% of all events the Imperfect Oracle generated predictions for; we added this error at the same rate for both benign and redteam events.

Ensemble Creation Engine

The Ensemble Creation Engine is used to create a new ensemble within the framework. It can create ensembles of two or more detectors that have been added to the framework. Because this Engine has minimal functionality and is tightly coupled with the Ensemble Engine, the two Engines were implemented in the same code space.

Ensemble Engine

This Engine is used to detect anomalies by combining the predictions of the individual anomaly detectors. There are several well-known algorithms to combine predictions, but

we decided to implement the "weighted average" method; doing so required coordinating with the Metrics Engine to calculate the AUROCC of the individual anomaly detectors. Each detector was weighted based on its AUROCC, and the predictions for each event were summed together to generate the final predictions.

Metrics Engine

The Metrics Engine is used to compute various metrics for the anomaly detectors and ensembles in the framework. To avoid duplicating code, this Engine computes the AUROCC for both the detectors and the ensembles. To do this, it first generates a ROC curve, then finds the area with trapezoidal integration.

Additionally, the Metrics Engine computes the diversity of the ensemble using the Q-Statistic. Because this metric was so closely tied to the ensemble, we implemented this Engine in the same code package as the ensemble and made it run automatically along with the Ensemble Engine.

Visualization Backend

The Visualization Backend was implemented as a Node.js web server. The Visualisation Backend is responsible for hosting the Visualization Frontend. For this MQP, the visualization was hosted on localhost; the analyst can access the visualization through their web browser at `http://localhost:[port number]`.

The Visualization Backend is also responsible for providing data to the Visualization Frontend. For the primary view, it must query the Aggregates table for the coarse aggregates and send data to the Frontend in the form of *(time, location1, location2, volume of events)*. Furthermore, the Backend must wait for the analyst to interact with the Visualization Frontend and supply data as needed. For the "isolate" view, the Backend must query the Aggregates table for the fine-grained aggregates and send these to the Frontend. For the "raw-data dump" feature, the Backend must query the set of events and return this data to the Frontend.

Visualization Frontend

The Visualization Frontend was implemented as a JavaScript web application using the D3 JavaScript library [12]. This library is dedicated to binding datasets to visual document objects on HTML web pages. For our purposes, D3 contained a number of advantages over well-known alternatives, such as Java applications developed with the Swing UI library. In particular, D3 is highly supported and well-known in the data visualization community for producing quick, scalable, and flexible tools. Mike Bostock, D3's creator, hosts a repository of many D3 projects and example code to use as guides for building new programs [11]. D3, as a JavaScript library, also benefits from the popularity and support of JavaScript

for developing data applications. Finally, D3 is a familiar language for developing mockup visualizations in Lincoln Lab, and our mentor Steven Gomez has used it to build visual demonstrations of laboratory projects.

The visualization displays the aggregated events as a heatmap, where each of the two axes is a *location*, or a field that represents *where the event occurred*. Example event locations include the source computer, user, subnet, geolocation, etc. In addition to the heatmap, the visualization supports a number of user interactions, such as isolating a value for one of the location Fields and investigate all events related to this value with respect to time or selecting a value for either the X or Y axes, or requesting a report containing the raw data for all events involving the selected value. For more information regarding the Visualization Frontend's final appearance and features, see Section 5.3: Experimentation for Hypothesis Two.

Chapter 5

Results

5.1 Interviews with Cybersecurity Analysts

We conducted three initial interviews with various analysts and researchers at the Lincoln Lab. We held our first interview with Person 1 (P1), a researcher in the Lincoln Lab Lincoln Research Network Operations Center (LRNOC) who has been working at the Lincoln Lab for 13 years. While she has not directly worked with data known to contain lateral movement, her experience has provided her with a solid background regarding the science of APTs and their detection. In particular, she has experience building data analysis and visualization tools, and could therefore provide useful feedback on our visualization design. Next, we interviewed Person 2 (P2), who has worked at the Lincoln Lab for seven years. He emphasized that his primary background involves computational algorithms rather than security, and he therefore based his responses on personal experience acquired from his work in LRNOC. We conducted a final interview with three analysts from the Lincoln Lab Information Services Department (ISD): Person 3 (P3), Person 4 (P4), and Person 5 (P5). Each of these analysts had a different level of experience at the Lincoln Lab, but they all primarily worked on detecting and mitigating threats to the Lincoln Lab network.

APTs and Lateral Movement Detection

During our first analyst interview with P1, we asked her to define lateral movement to ensure that we would be discussing the same concepts. Although she emphasized that a precise definition of lateral movement was difficult to pin down, we agreed to define it as the process of moving between computers maliciously, with the intent of spreading through the network to find high-profile hidden information. P1 said that APT detection is difficult because there are very few well-documented examples of APTs. Because of this scarcity (most events are *not* from an APT), it was inevitable that any analytic would have numerous false positives. However, LRNOC is extremely interested in reliable APT detection and has conducted some research in the area. LRNOC used many custom-built tools and techniques to facilitate lateral movement detection, typically using conservative anomaly-based detectors to prevent

the tool from generating excessive alerts. Despite these efforts, LRNOC did not yet have any anomaly detectors specifically for lateral movement.

P1's best advice for detecting lateral movement was to correlate multiple data sources, primarily authentication logs, email activity, DNS logs, and intranet activity. P1 referred to DNS logs as the "canary in a coal mine" because they appear in all network intrusions, no matter how careful the intruder. She also mentioned that emails, as major sources of malware, are typically heavily monitored in comparison to other types of network traffic. Another useful data source is intranet traffic (communication between individual computers), or at least cross-subnet traffic (communication between different computer groups), which would reveal any users who communicate with numerous computers or access a new host. P1 confirmed our authentication logs as a good place to begin lateral movement detection; however, many networks simply do not track enough information when monitoring authentications, and LRNOC cannot track all details of their authentication events. In addition, many of the logs collected are noisy, increasing the difficulty of distinguishing deliberate user actions from automated events. As such, correlating events from multiple sources across the entire detection lifecycle would prove more helpful than relying on the limited sources we obtained from the Los Alamos dataset.

P1 also described the lateral movement detection process from the analyst side in more detail. Once analysts observe odd behavior in authentication logs, P1 said, they would try to understand the context of the situation by investigating related logs for the activity of the relevant user or computer. An analyst in search of an APT must understand the context of the situation, including what other actions the user has performed, who else has performed actions on the relevant computers, and other details on how the user account and computer have been used. When we asked about APT mitigation, P1 named detection as the most challenging aspect of handling APTs, asserting that mitigation remained relatively simple provided that the compromised computer had been identified.

Our interview with P3, P4, and P5 confirmed P1's description of the situation and elaborated on the detection of cybersecurity threats at the Lincoln Lab. Although ISD does not specifically attempt to detect lateral movement or APTs, the analysts provided general advice on the matter. In particular, they explained that some events are only collected locally on individual machines and thus are not properly logged. Another issue with the detection of APTs of any kind is that, while they all have broadly similar motivations, they vary widely in design, strategy, and speed. P3 recommended reading reports of actual attacks to use as case studies, but warned us against basing our detection methods on the details of any one attack. The analysts also warned us that the dataset we used contains far more data than the typical network infrastructure is capable of collecting.

We inquired about the typical use of ools in ISD, but the exact details were confidential. However, we gleaned that analysts in ISD generally use signature-based rather than anomaly-based detection tools and rarely use visualizations other than prioritized lists of alerts. If they did want to detect lateral movement, they would probably use the tools they are already using.

Initial Visualization Feedback

When we consulted P1 for feedback on our initial visualization mockup (shown in Figure 5.1), she agreed that a heatmap showed promise as a visualization for lateral movement. She also confirmed that time, user, and source computer were the three most important axes for lateral movement detection. Her first thoughts were that the long diagonal line seemed normal, as each user will typically log into a single computer, and the vertical lines (representing multiple users on a single computer) also seemed normal if that computer was a server. The horizontal lines (representing a single user logging into multiple computers) were slightly concerning, as only an administrator would typically perform this activity. P1 confirmed that she would follow up on the activity represented by the horizontal lines by viewing the context of the users and computers in question.

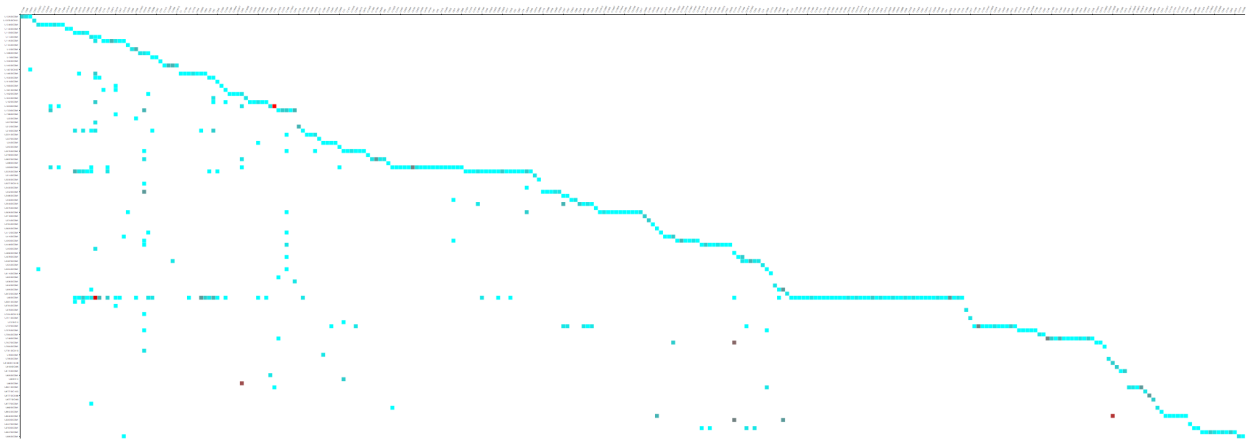


Figure 5.1: The mockup of our visualization that we showed to P1, P2, P3, P4, and P5. The columns represent destination computers, the rows represent source users, and the color of each block represents the concentration of events associated with the relevant user-computer pair. Cyan represents a low concentration of events, while red represents a high concentration.

P1 had several suggestions for various aspects of our visualization, including our method of ordering the axes, filtering, basic user interaction, and display of contextual information. Because our X-axis was ordered based on the first suspicious event, there was no way to infer network structure from the visualization. Ideally, the nodes would be ordered by network hierarchy; however, as this is explicitly not available in the Los Alamos dataset [27], a simple numerical numbering was acceptable. P1 advised against grouping computers by the frequency of their communications; this would result in a lack of continuity in computer order, which would be confusing for the analyst and would make comparison between timescales extremely difficult. The best sorting method, she said, would be one using static information (such as operating system, computer type, user role, etc.) to enable reliable comparison at different points in time.

P1 said that colors would only be relevant when they revealed a specific pattern, so color-coding the heatmap based on the most recent event in a user-computer pair was not helpful.

A color scale conveying the total number of events or their frequency would be more useful. P1 considered filtering based on the user or computer - perhaps to a subgraph of suspicious users and computers - to be a critical element of any visualization. A feature to allow the visualization to behave like a timelapse would also be useful. An essential use case, P1 said, was selecting a point to see all events at that point in the timeline, as well as allowing analysts to adjust a global timescale. Analysts always want to go back to raw data for investigation, so this should be easy for them. Along similar lines, our visualization would have to either allow anomaly detectors to be disabled or explain why any anomaly detectors flag events as suspicious. A lack of context for the final result is a well-known issue in machine learning, and it was especially important to consider in our visualization because any false positives could cause the analyst to lose their trust in the detector. P1 suggested including the anomaly detectors as an optional feature and allowing the analyst using the framework to build trust in them over time.

P2 had some further suggestions about the evaluation of our visualization, asserting that we must clearly rationalize our visualization to someone with a different background and develop appropriate coaching and training. In terms of the lateral movement detection itself, we needed to narrow our scope to focus on one type of lateral movement and refine our visualization to display relevant information in a clear and truthful manner. The visualization could not distort the data in any way unless we provided a clear indication of such a distortion and justification for its existence. At the same time, our visualization should require minimal effort on the analyst's behalf to run. Each feature we included should answer a question an analyst would have.

One major consideration of our visualization was the need to strike a balance between false positives and false negatives. False positives are an especially large problem in the realm of lateral movement detection, P1 explained, because of the relatively low frequency of such events. It is difficult to determine in advance whether to focus on reducing false positives or false negatives, but P1 estimated that an analyst could handle up to a few dozen alerts per day. For each alert, the analyst would have to follow up by returning to the raw data, analyzing the user, computer, history of the user/computer pair, authentication logs, and so on. For further feedback on our visualization, P1 suggested analysts in ISD because they work most closely with real data. Other LRNOC researchers could help refine our visualizations, but wouldn't necessarily be able to evaluate their usefulness.

P2 echoed many of P1's thoughts regarding false positives and false negatives: he said that analysts must negotiate both, and false positives, as long as they don't overwhelm the analyst, are generally acceptable. The analyst's most critical role is to understand and recognize normal conditions of network operations; precise metrics of false positives versus false negatives are less important than this overarching task. P2 posed a similar argument regarding anomaly detectors, which frequently present problems when applied to a visualization. The analyst loses trust in an analytic that does not provide explanations for why it flags certain events as suspicious, especially when flagged events are false positives. Following this statement, we decided that - despite its usefulness - anomaly detection data would likely detract from the analysis process if it was always visible in our visualization. Overall, P2 said, the analyst would remain focused on the typical state of the network

and would notice deviations from normal behavior more reliably than any single anomaly detector.

As part of a final interview to receive feedback on our visualization mockup, we once again presented our mockup to with P3, P4, and P5. Although these analysts work more closely with cybersecurity data than P1 and P2, they asserted that the only visualizations they regularly use are dashboards showing prioritized lists of alerts. The ISD analysts use visualizations primarily to satisfy their managers rather than for threat detection, and they saw little immediate use in our visualization proof-of-concept. For direct feedback, they emphasized maintaining simplicity in our visualization's design, as some analysts have only experienced "visualizations" created in Microsoft PowerPoint and Excel. Furthermore, they suggested highlighting the most alarming events (perhaps through changing size or color) to redirect the analyst's attention toward the most pressing matters. Additionally, the analysts expressed the desire to sort users and computers (e.g., grouping all administrators) and display the chain of events in an instance of lateral movement. For future talks with analysts, they added, we should clearly specify the framework's use cases: both the problem it solves and its relevant network situations. In addition, we should establish an explicit method of training analysts to use our framework to increase their understanding of the visualization's use cases.

When giving further advice, the analysts explained that our framework should be configurable (for example, allowing the analyst to specify which accounts are administrators). On the subject of false positives in signature-based detectors, they viewed the events as tedious to handle but not devastating for performance; as P3 said, "It's detecting what you tell it to detect. We don't lose trust in a tool, we just lose trust in how we told it to aggregate events." False positives are unavoidable, but as long as they are predictable, an analyst can develop a "profile" of normal false positives and will notice when the pattern of false positives differs from this profile. The potential issue with anomaly-based tools is that it may be difficult for the analyst to develop a profile of a tool that adapts based on the condition of the network data it is using.

5.2 Experimentation for Hypothesis One

This section describes the results of the experiment described in Section 3.3: Analytics and Evaluation of Hypothesis One, which we conducted to determine the validity of Hypothesis One as described in the Introduction. This section contains the results of preprocessing the data, the implementation of the Random Forest and Neural Network detectors, the performance of these two detectors, and the performance of the ensemble. Initially, we found that the ensemble did not have an acceptable diversity, and the Neural Network did not have an acceptable performance. Because of this, we repeated the experiment with a replacement detector to simulate an additional model.

Preprocessing the Los Alamos Dataset

We began the experiment by preprocessing the Los Alamos authentication dataset to remove all duplicate and incomplete events. After the initial preprocessing step, we isolated the densest window of redteam activity to use as our dataset. This window included 20 redteam events between times 764,106 and 765,017, reducing the size of the dataset from over one billion events to a mere 161,344 events. We then labeled these remaining events as *benign* or *malicious* based on the redteam dataset, using the assumption that the *only* malicious events were those initiated by the redteam, as stated in the Los Alamos dataset's documentation [**?**].

Initial Experiment Results

We executed the experiment for testing Hypothesis One as described in the Methodology chapter. Table 5.1 presents the results of this experiment; it shows the diversity of the ensemble in addition to the AUROCCs of the Random Forest detector, the Neural Network detector, and the ensemble.

Trial	Random Forest AUROCC	Neural Network AUROCC	Ensemble Diversity (Absolute Value)	Ensemble AUROCC
1	0.56	0.58	1	0.72
2	0.69	0.3	1	0.3
3	0.56	0.23	1	0.29
4	0.72	0.41	1	0.57
5	0.60	0.53	1	0.63
6	0.82	0.34	1	0.34
7	0.58	0.23	1	0.36
8	0.71	0.7	1	0.85
9	0.81	0.28	1	0.44
10	0.69	0.32	1	0.48
Average	0.68	0.39	1	0.5

Table 5.1: The results of the initial experiment used to test Hypothesis One. This table shows the AUROCCs for the Random Forest, the Neural Network, and the ensemble, as well as the ensemble diversity.

Because the average absolute value of the ensemble diversity metric (which represents the correlation between predictions) for the ensemble of the Neural Network and Random Forest was greater than our threshold of 0.75, it was not sufficiently diverse for this experiment. Because of this, we executed a revised version of the experiment with a replacement detector.

Revised Experiment Results

For the second iteration of this experiment, we implemented a new detector to replace the Neural Network, which had a lower average AUROCC than the Random Forest. Due to

time constraints, we used a modified version of the Oracle – called the *Imperfect Oracle* – rather than implementing a new machine learning model or refining our previous attempts. The Imperfect Oracle detected anomalies based on an event's ground truth label; however, it did so with a random error of 20%. The results of this repeated experiment are shown in Table 5.2.

Trial	Random Forest AUROCC	Imperfect Oracle AUROCC	Ensemble Diversity (absolute value)	Ensemble AUROCC
1	0.63	0.81	0.20	0.92
2	0.56	0.75	0.33	0.85
3	0.61	0.71	1	0.87
4	0.86	0.69	1	0.97
5	0.75	0.80	0.11	0.95
6	0.69	0.80	0.33	0.94
7	0.64	0.63	1	0.79
8	0.70	0.72	0.33	0.84
9	0.55	0.68	0.11	0.72
10	0.79	0.8	1	0.96
Average	0.68	0.74	0.54	0.88

Table 5.2: The results of the revised experiment used to test Hypothesis One. This table shows the AUROCC for the Random Forest, the Imperfect Oracle, and the ensemble, as well as the ensemble diversity.

5.3 Experimentation for Hypothesis Two

This section describes the results of the alternate experiment described in Section 3.4: Visualization and Evaluation of Hypothesis Two, which we conducted to determine the validity of Hypothesis Two as described in the Introduction. This section contains the results of preprocessing the dataset, a list of the visualization's views and user interactions, and the results from our final analyst evaluations.

Artificial Evaluation Dataset

To prepare the smaller dataset consisting of eight users (four of which were malicious) over 95 seconds, we preprocessed the original Los Alamos authentication dataset to remove duplicate and incomplete events. From here, we isolated all redteam events to find the densest period of unique redteam user activity. We decided to use the 95-second period between times 769,000 and 769,095, which contained redteam events from four users (U66@DOM1, U5254@DOM1, U8601@DOM1, and U1048@DOM1). We then selected four benign users (U8731@DOM1, U3451@DOM1, U640@DOM1, and U6146@DOM1) with activity in the same timespan to

contribute more events to the dataset. The resulting dataset contained 272 events, four of which were malicious.

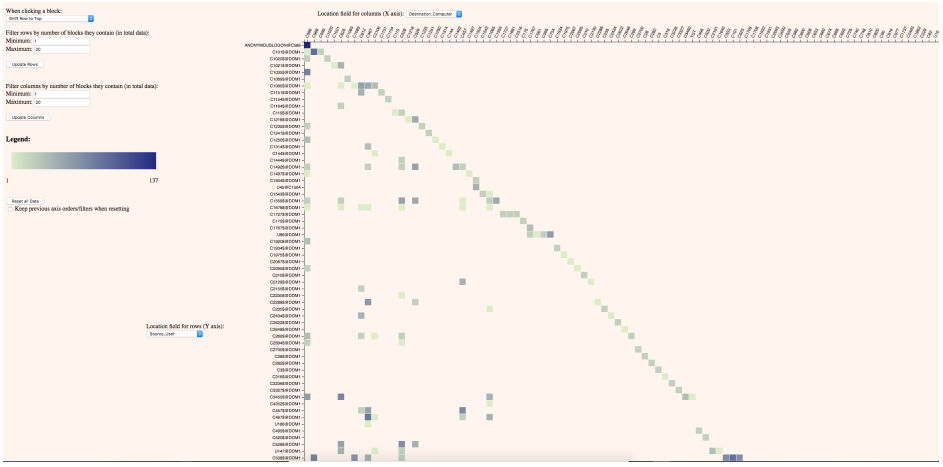
Visualization

Figures 5.2a and 5.2b show the primary visualization views for the two subsets of the Los Alamos dataset we used during the experiment. Figure 5.2a shows a "demo plot" of the first 10,000 events in the original Los Alamos dataset for familiarizing each subject with the visualization, and Figure 5.2b shows the plot of the eight test users plotted against destination computers. Figure 5.3 shows the timeplot – one of the main features of the visualization for discovering more information about the recorded users – which was generated from isolating User U22@DOM1's events in the demo. Finally, Figure 5.4 demonstrates the use of the Raw Data Dump feature on the Isolated Row screen, which allows the analyst to save a JSON log of the users they have isolated and each of their events.

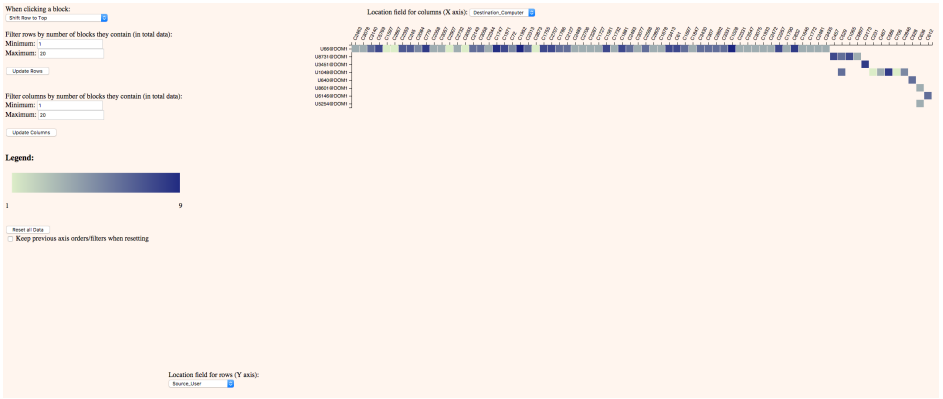
Feedback Session with Analysts

We conducted an informal interview with four analysts: P3, P5, P6, and P7. Overall, the analysts considered our proof-of-concept a good tool to isolate a set of users or computers worth exploring. According to one of the analysts we interviewed, it "would be a good indicator" of anomalous activity. The main benefit was the ability to leverage the insight of the human analyst in detecting suspicious activity. Analysts already rely on human deduction to efficiently evaluate threats; the purpose of our framework was to give the analyst the best perspective from which to do this. However, some aspects of it would need to change to best suit the analysts' needs. From the viewpoint of the analysts, our framework still presented the "raw data," which would take far too long to sift through. The estimated time to investigate even ten user accounts was between one and three weeks, which would quickly become unmanageable given the scale of most enterprise networks. One way to speed up this process would be to provide user information (especially phone number or other contact information) quickly and easily, but reducing the number of accounts that need investigation in the first place is critical.

The analysts would prefer to have the framework filter out the most important events automatically, then display a much smaller subset in our visualization. At a minimum, the framework should highlight the subsections of data that require further investigation; otherwise, the amount of raw data involved greatly increases the difficulty in locating behavior indicative of lateral movement. However, we confirmed the importance of minimizing the number of false alerts. This filtering will reduce the false positive rate and reduce the burden on the analyst. Analysts generally start to ignore the "criticality" (which we refer to as the suspicion level) of alerts if the tools that generate them tend to generate false positives. In addition, network heuristics tend to change on a weekly basis, which makes it difficult for detectors to keep up. As one analyst said, they're dealing with configuration errors "95% of the time." Analysts can usually tell whether an alert from a familiar signature-based anomaly



(a) Visualization with Los Alamos Demo Dataset



(b) Visualization with Experiment Dataset

Figure 5.2: The Visualization Frontend showing authentication events from (a) the 10,000 event demonstration dataset, and (b) the dataset used in Experiment 1b. Columns (configurable from dropdown menu above) represent the computers receiving authentications, and rows represent the user account which has authenticated with the specified computer. The color of an individual cell represents the relative volume of authentication events that occurred for the given computer and user account, where light green indicates a lower concentration of events and dark blue indicates a higher concentration of events.

detector is a false positive: alerts are usually not false positives if they break the typical pattern of the tool generating them. If our framework could not provide reliable anomaly detectors, the analysts suggested using some other indicator (such as failed authentication events) as a dimension for our visualization.

The analysts involved in our evaluation session had several other suggestions to improve the usability of our framework. They suggested the addition of sorting and organizing methods, especially ones that would highlight authentications at unusual times and distinguish between different types of users and computers. In addition, because analysts may not be familiar with the network they are monitoring, they would benefit from a display comparing current user activity to that of the average user in the network. This would provide the analyst with

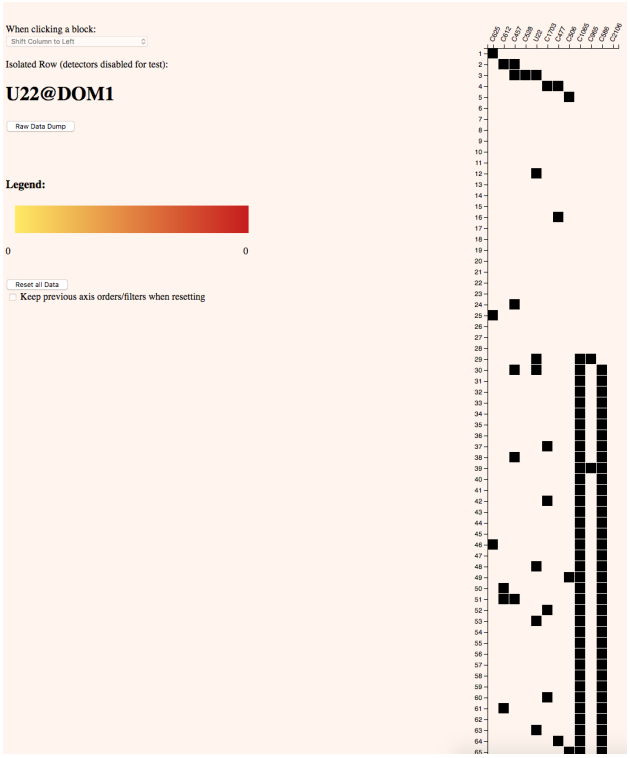


Figure 5.3: This figure shows the Isolated Time Plot feature of the visualization on the Los Alamos authentication events. When a block from a certain row is clicked, all events involving that row's location (default as Source User) are plotted against time. This figure shows the result of isolating U22@DOM1 (columns represent all computers into which the user authenticated, and rows represent the times in seconds during which these authentications occurred). During normal operation of the visualization, the blocks are shaded according to the Ensemble Engine's predicted suspicion level for the user/computer events at this time; however, to isolate the experiment as only comparing event time/locations without external aids, we left the blocks unshaded.

a benchmark of user behavior and assist them in more accurately determining when a given account is suspicious.

In addition, our framework must display data from multiple sources to be useful; otherwise, it would not effectively capture the whole picture. Correlating logs from multiple sources is especially important because authentication logs are noisy and unpredictable: some events may be dropped or duplicated an unknown number of times. Other data would include information on the user accounts (admin status, password changes, time zone, etc.) and computers (server vs. client, etc.). Although we would not necessarily have access to this in real-world data, it was important to support it if at all possible. Adding the data to the summary displayed while hovering over a cell would prove especially helpful.

```
U3@DOM1
[
   [
      {
        "Event": 169,
        "Time": 1,
        "Name": "Destination_Computer",
        "FieldValue": "C419"
      }.
      {
        "Event": 169,
        "Time": 1,
        "Name": "Logon_Type",
        "FieldValue": "Network"
      },
      {
        "Event": 169,
        "Time": 1,
        "Name:": "Source_User",
        "FieldValue": "U3@DOM1"
      },
      {
        "Event": 169,
        "Time": 1,
        "Name": "Success_Or_Failure",
        "FieldValue": "Success"
      },
      {
        "Event": 169,
        "Time": 1,
        "Name": "Source_Computer",
        "FieldValue": "C419"
      },
      {
        "Event": 169,
        "Time": 1,
        "Name": "Authentication_Type",
        "FieldValue": "?"
      },
      {
        "Event": 169,
        "Time": 1,
        "Name": "Authentication_Orientation",
        "FieldValue": "Logoff"
      },
      {
        "Event": 169,
        "Time": 1,
        "Name": "Destination_User",
        "FieldValue": "U3@DOM1"
      }
   ]
```


Figure 5.4: This figure shows the Raw Data Dump feature of the visualization on the Los Alamos authentication events. The value U3@DOM1 was selected for the user account, and the report shows the entire set of authentication events that occurred from User U3@DOM1 in the dataset being used.

Evaluation with Analysts

We presented the eight-user experimental dataset in our visualization, described as containing between one and seven redteam users utilizing lateral movement, to four subjects in an informal evaluation. We asked each subject to examine the details of each user's data before classifying them as either "malicious" or "benign." The results from this experiment are

shown in Figures 5.5 and 5.6, which display both the raw results of each analyst's predictions and an analysis of their success and error rates.

Username	U66	U5254	U8601	U1048	U8731	U3451	U640	U6146
S1 Predictions	Malicious	Benign	Benign	Benign	Benign	Benign	Benign	Benign
S2 Predictions	Benign	Benign	Benign	Benign	Malicious	Benign	Benign	Benign
S3 Predictions	Benign	Benign	Benign	Malicious	Malicious	Benign	Benign	Benign
S4 Predictions	Malicious	Benign	Benign	Malicious	Malicious	Benign	Benign	Benign
Ground Truth	Malicious	Malicious	Malicious	Malicious	Benign	Benign	Benign	Benign

Figure 5.5: The analysts' results from classifying the users in the experimental Los Alamos dataset as either *malicious* or *benign*.

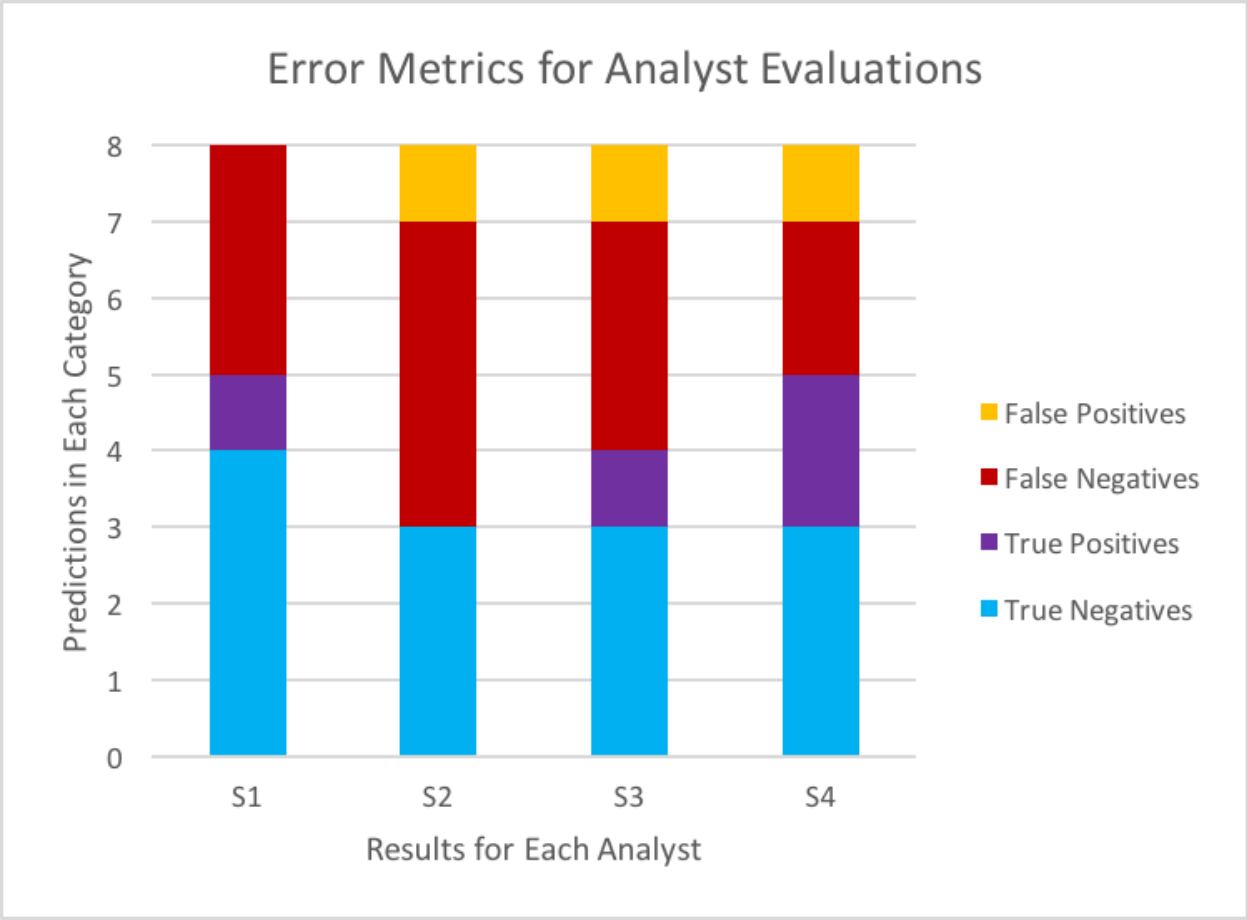


Figure 5.6: The analysts' results from classifying the users in the experimental Los Alamos dataset, shown in terms of success and error metrics.

Chapter 6

Conclusion

6.1 Discussion of Hypothesis One

Because the ensemble did not meet our required metrics for the execution of Experiment 1a, we decided to repeat the procedure with a different ensemble configuration in Experiment 1b. Because the ensemble diversity in our experiment was extremely poor, and the AUROCC of one of the detectors was below 0.5, our Experiment 1a broke the assumptions that ensemble learning is based on. An improvement over the average AUROCC with an ensemble is only expected when the error of each component detector is below 50%, which was not the case with Experiment 1a. Given this situation, we did not expect our ensemble's AUROCC to be an improvement over those of the two component detectors. Indeed, the ensemble's AUROCC of 0.497 was significantly lower than the Random Forest's AUROCC of 0.675, and was even lower than the average detector AUROCC of 0.5335. As such, we could not use the Experiment 1a results of this ensemble to evaluate Hypothesis One.

In order to correct this, we replaced the Neural Network - which had the lower AUROCC of the two detectors - with an improved detector. The Neural Network most likely performed poorly because it only considered the *Authentication Type*, *Logon Type*, *Authentication Orientation*, and *Success or Failure*. These features were likely less indicative of lateral movement than the features used by the Random Forest detector (*Source User*, *Source Computer*, *Destination User*, and *Destination Computer*). Unfortunately, applying these four features to the Neural Network would not have been a trivial task; since neural networks create a new input node for all possible values of the input features (see Section 4.3: Framework Components), adding these four features would have resulted in a neural network with more than 30,000 input nodes. Such a large number of input nodes would have introduced noise into the model, likely reducing its accuracy rather than improving it.

Thus, instead of improving the Neural Network detector, we developed a new detector: the Imperfect Oracle detector. The Imperfect Oracle was a modified version of the Oracle that assigned an incorrect value to 20% of events. Ideally, we would have used a detector that did not use the ground truth labels to test our ensemble in a more "realistic" scenario; however,

due to time constraints, we decided that it would be infeasible to implement a third non-trivial detector. Additionally, because Hypothesis One was independent of the details of the component detectors, implementing this trivial detector did not reduce the experiment's ability to validate the hypothesis.

Because the detector AUROCCs were both greater than 0.5 and the ensemble's diversity (0.54) was less than the upper bound of 0.75, the experiment using the Random Forest and Imperfect Oracle was a valid test of Hypothesis One. Thus, our evidence supported Hypothesis One because the average ensemble AUROCC across the 10 trials (0.881) was larger than the AUROCCs of the Random Forest (0.677) and the Imperfect Oracle (0.738). In fact, the ensemble had a significantly larger AUROCC than that of its best component detector, rather than merely outperforming the average AUROCC, which demonstrated that the diversity was far enough from the threshold for the ensemble to see great improvements over the individual detectors.

6.2 Discussion of Hypothesis Two

Informal Analyst Evaluation

Although our existing proof-of-concept framework was not immediately usable for lateral movement detection, the concept has the potential to be useful for this purpose. Because our implemented framework was merely a proof-of-concept, we did not expect any of our code to be used in future work; however, the concepts and results are capable of informing future efforts in similar directions. We found several aspects of our framework that could be improved in addition to new features that could be implemented.

The most important improvements are ones that increase the chances for an analyst to isolate subsets of the data to investigate. One potential improvement would be some sort of preprocessing to remove all but the most relevant events. Because the datasets we used are far too large to view in a single visualization, and the timespans involved are measured in days or weeks rather than seconds or minutes, a sparse display of the most suspicious events would be significantly more helpful to the analysts than a simple display of all the raw data involved. Although lateral movement may have been visible at small scales in certain configurations of our visualization, none of us were able to isolate redteam activity even in relatively small datasets that included all events in a time range greater than a few seconds. Another enhancement to enable the analyst to isolate suspicious activity would be a data fusion method that effectively combines authentication logs with logs of other processes. Although correlating process logs to authentication logs would be invaluable when determining which users are conducting which activities, this is a difficult problem if logoffs are not recorded (which is the case in most networks). Although our framework was designed to be extensible and make use of multiple data sources, this problem must be solved before the framework can reach its full potential. A smaller - but still useful - feature would be a display of the "average" user activity to use as a benchmark to define normal behavior. Because many analysts do not know what the typical behavior looks like

for a given network, an example would be invaluable for determining whether a given user is conducting suspicious behavior.

The analysts we interviewed also had several suggestions for more small-scale (but nonetheless important) features to implement. A method for implementing additional sorting and filtering techniques would be a helpful way to fit the requirements of individual networks. The most important of these is a sorting method that accounts for unique users or computers in authentication events, rather than a simple count of all the authentication events associated with the given user or computer. One extension suggested by our framework's structure is the addition of signature-based anomaly detectors to improve reliability and analyst control. This would be relatively easy to implement, as our framework is designed to enable the addition of arbitrary detectors as plugins. This adaptation, though simple, is important to mention because it would make the framework much more understandable and trustworthy to the analysts using it. Ideally, the framework would display justifications for detector estimates of suspicion levels for individual events. Several of our sources have confirmed that analysts will not fully trust alerts if they do not know the reasoning behind them, so this feature would enable the analyst to determine whether the justification is valid without needing to investigate further. This would be much easier to implement with signature-based detectors than anomaly-based detectors, though it could presumably be done using either method.

The analyst could also make use of timezone and other time-related information to highlight unusual user activity in terms of time. Because many APTs operate from other time zones, this would be a relatively simple way to detect APTs if the network being protected is largely centered in a single timezone. Displaying a more complete summary to the analyst when they hover over a square of the heatmap would also make anomalous activity easier to see. This information could be difficult to visualize directly, but it may still be essential in determining whether a given account is involved in lateral movement. Displaying this information in a summary of the currently selected square of the heatmap visualization is the most efficient and intuitive way to convey it to the analyst.

Experiment Results

While our original plan for an experiment on the visualization required having analyst subjects distinguish between two subsets of the Los Alamos dataset, one containing redteam events and the other containing no such events, we realized that this plan was infeasible due to the size of the datasets it would require. In the past, we had primarily tested the visualization on a sample dataset containing 10,000 events (the same one used for our alternate experiment's demonstration phase). Because the visualization remained responsive when using this dataset (with a lag time of less than half a second at most), we did not realize that our new datasets would exceed the visualization's technical capabilities. Indeed, even our smallest redteam dataset of 160,000 events required approximately 1 minute of loading, and all attempts to scroll or interact with the visualization in this state resulted in significant lag time.

While we typically did not prioritize usability for the sake of our hypothesis-driven project, we agreed that asking analysts to participate in an evaluation using an unresponsive visualization would be unfair to the analysts involved and would inevitably produce poor results. For this reason, we redesigned the experiment to operate on a much smaller dataset that could easily render in our visualization, allowing the subject to minimize overhead in operating the visualization and quickly see the results of their operations. Given more time, we would have improved our visualization's technical efficiency, allowing us to conduct the original experiment in order to observe the visualization's effectiveness on a more realistic dataset compared to a very filtered set of explicit redteam users. However, as we performed the alternate experiment within the final two weeks of the project, such efficiency improvements would fall outside the scope of our project and its time constraints.

In the final results of our alternate experiment, we discovered that our visualization's arrangement of network data by time and location pairs did not provide the analyst with a strong aid for identifying anomalous activity. On a grade from 0 to 8 of correctly labeled users, each subject scored within one point of 4/8 (or 50% accuracy), with a maximum of only two out of the four redteam users labeled as malicious. Multiple subjects noted the difficulty in identifying malicious users with only one or two events through the timeset (i.e., U8601@DOM1 and U5254@DOM1), who appeared on the visualization as ostensibly benign users simply logging into a single computer. While the subjects produced a relatively low number of false positives in their classification, only 1/4 at most, the higher rate of false negatives indicates that this proof-of-concept visualization cannot be used to reliably isolate malicious users from benign ones. However, one major limitation of this evaluation was that many analysts found our proof-of-concept visualization difficult to use, and at least half of them forgot that they could change the axes on the heatmap. Although all members of the team could see a clear difference between redteam and benign users in a certain combination of dimensions, most of the analysts involved in the evaluation did not see this view because of the difficulty of changing dimensions in our proof-of-concept. Thus, it was unclear whether our ability to determine which users were involved in redteam activity was due to the clarity of our visualization or merely our familiarity with the Los Alamos dataset.

Through analysis of the qualitative feedback from analysts and the experiment results outlined above, we have determined that our evidence did not support Hypothesis Two: our visualization of times and locations from authentication network records did not produce a visible pattern of lateral movement between nodes. However, this conclusion does not invalidate the visualization technique's usefulness for presenting network data to analysts. Rather, it simply exposes the limitations of our project's scope and highlights the need for continued analyst support in network monitoring, record collecting, and visualization tool development in order to improve lateral movement detection. As multiple analysts highlighted during our interviews, the Los Alamos authentication dataset conveys heavily limited information, containing no data regarding a computer's IP address, subnet grouping, or criticality, and the logins lack details regarding their numeric authentication type (frequently used to track pass-the-hash attacks). Since analysts will frequently search for anomalies in these additional fields when attempting to determine a user's authenticity, a visualization that supports filtering or sorting datasets with this supplemental information is far more likely to quickly and accurately direct the analyst's attention to the network's most

suspicious users. By combining support for supplemental authentication record information with the visualization's ability to quickly display clusters or outliers in the rows and columns, the visualization will continue to direct analysts toward interesting parts of the dataset while presenting more heuristics to accurately pinpoint malicious intent in the events.

6.3 Future Work

This section discusses the aspects of this MQP that we did not address due to time and resource constraints and poses them as topics for future exploration. In addition, we will also cover future opportunities for usage of our framework at the Lincoln Lab.

Our framework's database architecture stored all data points as character strings. This allowed the schema to be generalized to fit most cybersecurity datasets; however, it simultaneously reduced the dimensionality of the data. This reduction of dimensions may have eliminated valuable information; in the case of an IP address, storing values as strings would not allow the data to represent the network topology, which could be valuable information. It would be worthwhile to explore database schemas that remain flexible for a variety of cybersecurity datasets, yet allow for more sophisticated data representation.

The framework's implementation of many algorithms, data structures, and database queries were inefficient, which limited the volume of data that the framework could process. If these were optimized to allow the framework to handle larger volumes of data, the performance of the individual detectors (and the ensemble) would likely improve due to the increased volume of training data. This improvement would likely allow our Random Forest Detector to reach an AUROCC near 0.9, as described in the paper *Using Bipartite Anomaly Features For Cyber Security Applications* [19].

To enable the detector plugins to be language-agnostic, we executed their code using Java's *exec* command. Doing so introduced a tremendous security risk; it allowed for the execution of arbitrary code. If our framework were to be used by anyone outside of this MQP team, this vulnerability would need to be addressed. One option to mitigate this risk is to *sandbox* the detectors. Sandboxing is the practice of separating the execution of a process on a machine from all other processes, with the goal of limiting the malicious capabilities of the sandboxed process. This practice could be implemented to reduce the vulnerability of the framework, but a more thorough investigation of the framework's security would still be necessary.

Opportunities at MIT Lincoln Laboratory

The Lincoln Lab and its sponsors have expressed interest in conducting research into lateral movement, and as a project designed to examine the capabilities for analytics and visualizations to detect lateral movement, our framework will most likely complement and support their efforts in this endeavor. In particular, members of Group 58 (Cyber Analytics and Decision Systems) have developed a tool known as Dynamic Flow Isolation (DFI), which leverages technology in software-defined networking systems to dynamically control access

between hosts and enforce the "principle of least privilege," which restricts users from accessing resources that lie outside the requirements of their role. One of DFI's main applications is the protection of its networks against lateral movement, made possible by removing full network connectivity and only permitting users to access required business resources on the network. As the network dynamically adapts to logon events and sensor data to establish and enforce user privilege, the rule-based network controllers will automatically reject user attempts to move laterally into unauthorized hosts.

Considering DFI's use as a tool to hinder lateral movement on networks, our project framework has the strong potential to support and verify DFI's performance in this regard. During redteam exercises on DFI-protected networks, our framework could potentially ingest datasets of the network events and output them into the detection and visualization components, allowing the testing analysts to view a succinct summary of network activity for the exercise. In this manner, our framework will allow the analyst to observe DFI's success in preventing redteam lateral movement: for events occurring when DFI has been enabled, the analyst can use the visualization to verify the lack of clear, malicious lateral movement within the network. The analyst can then compare these results to another set of events corresponding to when DFI has been disabled, in which he or she would expect to see evidence of unauthorized network movement spreading across hosts in the visualization's patterns. Conversely, DFI could also support attempts to acquire "clean" datasets (ones lacking lateral movement) for testing the framework; logging events on DFI-protected networks would greatly reduce the chance of mistakenly adding data with unauthorized movement to a ground truth dataset.

In general, as MIT Lincoln Laboratory continues to research lateral movement and strategies for its mitigation, our framework could provide useful support for validating the performance of current and future tools designed for this purpose. Similarly, as methods to detect or isolate lateral movement continue to gain momentum in upcoming years, the framework can leverage these strategies to provide more accurate detections and displays of lateral movement in its visualization. As such, while our framework will require continued development to increase its effectiveness at pinpointing malicious events, the project can establish a mutually symbiotic relation with lateral movement prevention tools within the Lincoln Lab, catalyzing and accelerating the research process into lateral movement as a whole.

Appendix A

Requirement Interview Questions

The following is the default set of Requirement interview questions used to obtain information regarding intrusion detection, APTs, lateral movement, and cyber visualization from analysts in our first meeting with them. Interviews with analysts were in a semi-structured format, and each conversation focused primarily on the analyst's greatest area of expertise within the domain of our project. As such, not every interview strictly followed the questions outlined below; we gave increased attention to questions particularly relevant to a particular analyst's background, and only devoted limited time to questions about which the analyst had less experience.

1. What is your job at Lincoln Laboratory?
2. How long have you been working at Lincoln Laboratory?
3. How much experience do you have in intrusion detection and cybersecurity visualization?
4. Do you believe Advanced Persistent Threats (APTs) should be a large concern for everyday cybersecurity (at Lincoln Laboratory or outside)?
5. Are you familiar with the concept of Lateral Movement by an APT?
   (a) If so, how would you precisely define the term?
6. Does your work include detecting APTs and/or Lateral Movement, or other anomalous network behavior?
   (a) If so, what are the tools and techniques you typically use for this?
   (b) If not, what tools and techniques might you try to use when you start?
   (c) What tools and techniques are typically used for this by most cybersecurity analysts?
7. In regards to the aforementioned tools:
   (a) What works particularly well for helping to detect APTs?

(b) What are the greatest challenges and shortcomings in the intrusion detection process?

8. Which phase in the killchain are analysts and defenders most likely to detect and stop an APT?
9. If an APT or lateral movement is detected while still active, what information within the data is necessary to know in order to mitigate and/or eliminate the threat?
10. What patterns in data do analysts search for when trying to find evidence of lateral movement (or an APT in general)?
11. Do cybersecurity analysts generally use data visualization tools to investigate patterns in large datasets in their cybersecurity work?
    (a) Could these data visualizations be relevant to the field of intrusion detection and locating APTs?
12. Do you have any ideas for features that would improve your detection process or general ease of use in a visualization?
13. Are 2D matrix/heatmap-based visualizations commonly used in cybersecurity, or are these visualization types relatively rare?
14. [after explaining the intuition behind our visualization concept] Do you believe the assumptions we have outlined about the threat are reasonable? Do you think you could make sense out of the visualization we have proposed, if it were built and loaded with data? Are there any other types of visualization that would likely be more conducive to identifying lateral movement?
15. Do analysts feel fatigue from handling false positive alerts from the tools or processes they use?
    (a) In general, is the false positive rate a more significant problem than the false negative rate?
16. When analysts receive an alert about anomalous or malicious behavior, how do they validate whether the alert describes a genuine problem?
17. Do analysts look at the output of different methods and tools in order to validate these alerts?
    (a) If so, what are the strengths and weaknesses of combining different analytics' methods and tools through the intrusion detection process?

Appendix B

Literature Review Search Terms

The most common search terms we used for our literature review were:

- Lateral movement with additional terms:
  - APT
  - Computer
  - Detection
  - Network
  - Threat
  - Splunk
  - BRO
  - Ikram Ullah
- Visualization with additional terms:
  - Network
  - APT
  - Detection
  - Cyber
  - Human-centered design
  - Anomaly
- Combinations of visualization and lateral movement, along with the terms:
  - IP address
  - Cybersecurity

  - Attack graph
  - External
- SIEM (Security Information and Event Management) with additional terms:
  - Security
  - Lateral movement
- Pass the hash
- Ensemble learning with additional terms:
  - Data fusion
  - Machine learning
  - Diversity
  - Methods

In addition to these search terms, we relied on the references of useful papers to access related sources. We prioritized reading papers that were cited multiple times because they were more likely to be reliable and helpful sources.

Glossary

Analyst A cybersecurity expert capable of detecting and managing threats and intrusions in their organization's network. i, iii–v, 3–5, 8–10, 13, 14, 16, 18, 21–24, 31, 32, 34–37, 44–46, 48, 50–54, 56–58, 61, 62, 64–68

Anomaly-Based Strategy An intrusion detection strategy that establishes a profile of normal user behavior and flags events that deviate from these profiles. 3, 9, 10, 14, 21, 50, 51, 54, 65

APT Advanced Persistent Threat. i, ii, iv, v, c, 1–3, 6–10, 16, 19, 21, 23, 24, a, b, f, 50, 51, 65

AUROCC Area Under the Receiver Operating Characteristic Curve. iv, v, 3–5, 11, 29, 30, f, 40, 41, 48, 55, 56, 63, 64, 67

Cyberattack An illegal attempt to harm someone's computer system or the information on it, generally with the use of the Internet. iv, 1, 6, f

Cybersecurity Measures taken to protect a computer or computer system (as on the Internet) against unauthorized access or attack. i–iv, c, 3, 4, 10, 13, 19, 20, 23–25, a, b, e, 32, 34, 35, 43, 46, 51, 54, 67

Cyberspace The environment in which communication over computer networks occurs. 1–3

Ensemble A collection of detectors whose individual predictions are combined into a single, more accurate prediction. Most commonly used in the context of machine learning. i, iv, v, d, 3–5, 12, 13, 23, 26, 28–31, 38, 40–42, 45–48, 54–56, 59, 63, 64, 67

Ensemble Diversity A key metric in ensembles which encapsulates the systematic differences in predictions between detectors. Diverse detectors will make different kinds of errors and thus be more diverse. viii, 12, 29–31, 55, 56, 63, 64

False Negative Rate The rate at which events in a prediction are incorrectly classified as negative. It is calculated with $\frac{FN}{FN+TP}$. 3, 22

False Positive Rate The rate at which events in a prediction are incorrectly classified as positive. It is calculated with $\frac{FP}{FP+TN}$. iv, 3, 10, 11, 22, 29

Hypothesis One An anomaly detector ensemble that combines the predictions from a sufficiently diverse set of component anomaly detectors will detect malicious events with an AUROCC that is greater than or equal to the average AUROCC of its component detectors. 3, 5, 25, 29, 31, 54–56, 63, 64

Hypothesis Two A visualization that organizes anomalous events based on each event's time, location, and suspicion level will allow an analyst to determine whether a given pattern of events is caused by lateral movement. 4, 5, 23–25, 31, 34–36, 56, 66

Intrusion Detection the process of monitoring the events occurring in a computer system or network and analyzing them for signs of possible incidents. 3, 9, 10, 20, 24

ISD Information Services Department. 24, 34, 50, 51, 53, 54

Lateral Movement The process of moving within a network to gain further access and control. One of the typical steps of an APT. i, iv, v, c, d, 3–5, 7–9, 14, 16, 18, 19, 21–24, 31, 32, 34, 36, 37, 50–54, 57, 60, 63–68

Lincoln Lab MIT Lincoln Laboratory. iii–v, 23, 24, 31, 32, 35, 49–51, 67, 68

LRNOC Lincoln Research Network Operations Center. 24, 50, 51, 53

Machine Learning The cross-section between computer science, statistics, and signal processing that aims to develop programs capable of learning a specific functionality based on preexisting data. iv, v, 3–5, 10–12, e, 31, 47

MQP Major Qualifying Project. i, iii, 3, 4, 11, 19, 26, 39, 47, 48, 67

NIST National Institute of Standards and Technology. 9

P1 Person 1. 50–54

P2 Person 2. 50, 52–54

P3 Person 3. 50–52, 54, 57

P4 Person 4. 50–52, 54

P5 Person 5. 50–52, 54, 57

P6 Person 6. 57

P7 Person 7. 57

Redteam A reference to the practice of "red versus blue" cyberattack and cyberdefense training exercises. "Redteam" is generally synonymous with "malicious". 19, 20, 24, 25, 30, 32, 33, 35, 36, 47, 55, 56, 60, 64–66, 68

ROC Receiver Operating Characteristic. vii, 11, 29, 48

SCGDM Suspiciousness Cascading Graph Detection Method. 21–23

SCP Secure Copy Protocol. 2

Signature-Based Strategy An intrusion detection strategy that encodes *signatures*, or information about behaviors that are known to be indicative of intrusion, and filters events based on these signatures. 3, 9, 10, 14, 21, 22, 51, 54, 57, 65

VM Virtual Machine. 2

Bibliography

[1] Service Departments. *MIT Lincoln Laboratory*.

[2] *Understanding Intrusion Detection Through Visualization*, pages 1–14. Springer US, Boston, MA, 2006.

[3] National CyberWatch Mid-Atlantic Collegiate Cyber Defense Competition. NETRESEC, 2012.

[4] ROC Curve Analysis. MEDCALC, 2017.

[5] Threat Hunting Reference Guide. Sqrrl Threat Hunting Blog, `https://sqrrl.com/threat-hunting-reference-guide/`, 2017.

[6] Visualization Security Data. VizSec, `http://vizsec.org/data/`, 2017.

[7] J. Anderson. Computer Security Threat Monitoring and Surveillance. Technical Report 79F296400, National Institute of Standards and Technology, Box 42 Fort Washington, PA. 19034, Feb 1980.

[8] R. Bace. *Intrusion Detection*. Macmillan Technical Publishing, 201 West 103rd Street, Indianapolis, IN 46290, 2000.

[9] R. E. Banfield, L. O. Hall, K. W. Bowyer, and W. P. Kegelmeyer. Ensemble Diversity Measures and Their Application to Thinning. *Information Fusion*, 6(1):49 – 62, 2005. Diversity in Multiple Classifier Systems.

[10] D. M. Best, A. Endert, and D. Kidwell. 7 Key Challenges for Visualization in Cyber Network Defense. In *Proceedings of the Eleventh Workshop on Visualization for Cyber Security*, VizSec '14, pages 33–40, New York, NY, USA, 2014. ACM.

[11] M. Bostock. About Blocks. bl.ocks.org, Sep 2010.

[12] M. Bostock, V. Ogievetsky, and J. Heer. D3: Data-Driven Documents. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)*, 2011.

[13] P. Chen, L. Desmet, and C. Huygens. A Study on Advanced Persistent Threats. Number 8735, 2014.

[14] B. Ewaida. Pass-the-Hash Attacks: Tools and Mitigation. *SANS Institute InfoSec Reading Room*, pages 2–51, Jan 2010.

[15] A. Fawaz, A. Bohara, C. Cheh, and W. Sanders. Lateral Movement Detection Using Distributed Data Fusion. In *2016 IEEE 35th Symposium on Reliable Distributed Systems (SRDS)*, pages 21–30, Sept 2016.

[16] T. Fawcett. An Introduction to ROC Analysis. *Elsevier*, Dec 2005.

[17] S. Garcia. CTU-13. Stratosphere IPS, `https://stratosphereips.org/category/dataset.html/`, 2015.

[18] B. Glithero. Better Threat Detection and Response with Analytics for Lateral Movement. Pivotal, `https://content.pivotal.io/blog/better-threat-detection-and-response-with-analytics-for-lateral-movement/`, 2017.

[19] E. Goodman, J. Ingram, S. Martin, and D. Grunwald. Using Bipartite Anomaly Features for Cyber Security Applications. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pages 301–306, Dec 2015.

[20] A. Hagberg, A. D. Kent, N. Lemons, and J. Neil. Credential Hopping in Authentication Graphs. In *2014 International Conference on Signal-Image Technology Internet-Based Systems (SITIS)*. IEEE Computer Society, Nov. 2014.

[21] J. Heaton. Encog: Library of Interchangeable Machine Learning Models for Java and C#. *Journal of Machine Learning Research*, 16:1243–1247, 2015.

[22] C. Hummel. Why Crack When You Can Pass the Hash? *SANS Institute InfoSec Reading Room*, pages 2–39, Oct 2009.

[23] T. Itoh, H. Takakura, A. Sawada, and K. Koyamada. Hierarchical Visualization of Network Intrusion Detection Data. *IEEE Computer Graphics and Applications*, 26(2):40–47, March 2006.

[24] E. Jones, T. Oliphant, P. Peterson, et al. SciPy: Open source scientific tools for Python, 2001–.

[25] N. Kawaguchi, H. Tomimura, and M. Tsuichihara. Building Suspiciousness Cascading Graph over Multiple Hosts for Detecting Targeted Attacks. In *2016 IEEE International Conference on Communications (ICC)*, pages 1–6, May 2016.

[26] R. Kemmerer and G. Vigna. Intrusion Detection: A Brief History and Overview. *Computer*, 35(4):27–30, Apr 2002.

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

[28] A. D. Kent. Cybersecurity Data Sources for Dynamic Network Research. In *Dynamic Networks in Cybersecurity*. Imperial College Press, June 2015.

[29] A. D. Kent, L. M. Liebrock, and J. C. Neil. Authentication Graphs: Analyzing User Behavior Within an Enterprise Network. *Computers and Security*, 48:150 – 166, 2015.

[30] L. Kuncheva and C. Whitaker. Measures of Diversity in Classifier Ensembles and Their Relationship With the Ensemble Accuracy. *Machine Learning*, 51(2):181–207, May 2003.

[31] Q. Liao. Visual Analytics for Network Security and Anomaly Detection. Central Michigan University Department of Computer Science, `http://people.cst.cmich.edu/liao1q/VizSec.shtml`, Jul 2017.

[32] M. Lichman. UCI Machine Learning Repository. University of California, Irvine, School of Information and Computer Sciences, 2013.

[33] W. McKinney. pandas: a Foundational Python Library for Data Analysis and Statistics. 2011.

[34] R. Nolette. Finding Evil When Hunting for Lateral Movement. Sqrrl Threat Hunting Blog, `https://sqrrl.com/finding-evil-when-hunting-for-lateral-movement/`, May 2017.

[35] E. Novikova and I. Kotenko. Analytical Visualization Techniques for Security Information and Event Management. In *2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, pages 519–525, Feb 2013.

[36] E. Novikova and I. Kotenko. Analytical Visualization Techniques for Security Information and Event Management. In *2013 21st Euromicro International Conference on Parallel, Dstributed, and Network-Based Processing*, pages 519–525, Feb 2013.

[37] A. Oberle, P. Larbig, R. Marx, F. Weber, D. Scheuermann, D. Fages, and F. Thomas. Preventing Pass-the-Hash and Similar Impersonation Attacks in Enterprise Infrastructures. In *2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)*, pages 800–807, March 2016.

[38] A. Oberle, P. Larbig, R. Marx, F. G. Weber, D. Scheuermann, D. Fages, and F. Thomas. Preventing Pass-the-Hash and Similar Impersonation Attacks in Enterprise Infrastructures. In *2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)*, pages 800–807, March 2016.

[39] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[40] R. Polikar. Ensemble Based Systems in Decision Making. *IEEE Circuits and Systems Magazine*, 6(3):21–45, Third 2006.

[41] S. M. Sawyer, T. H. Yu, M. L. Hubbell, and B. D. O'Gwynn. LLCySA: Making Sense of Cyberspace. *Lincoln Laboratory Journal*, 20(2):67–76, 2014.

[42] K. Scarfone and P. Mell. Guide to Intrusion Detection and Prevention Systems (IDPS). Technical Report 800-94, National Institute of Standards and Technology, Gaithersburg, MD 20899-8930, Feb 2007.

[43] M. Sconzo. Samples of Security Related Data. SecRepo, `http://www.secrepo.com/`, 2017.

[44] H. Siadati, B. Saket, and N. Memon. Detecting Malicious Logins in Enterprise Networks Using Visualization. In *2016 IEEE Symposium on Visualization for Cyber Security (VizSec)*, pages 1–8, Oct 2016.

[45] P. Sollich and A. Krogh. Learning with Ensembles: How Overfitting Can be Useful. In *Advances in Neural Information Processing Systems*, pages 190–196, 1996.

[46] C. Tankard. Advanced Persistent Threats and How to Monitor and Deter Them. *Network Security*, 2011(8):16 – 19, 2011.

[47] I. Ullah. Detecting Lateral Movement Attacks through SMB using BRO. Technical Report, `http://essay.utwente.nl/71415/`, November 2016.

[48] S. Wagh, V. Pachghare, and S. Kolhe. Survey on Intrusion Detection System Using Machine Learning Techniques. *International Journal of Computer Applications*, 78(16), 2013.

[49] K. Woods, W. Kegelmeyer, and K. Bowyer. Combination of Multiple Classifiers Using Local Accuracy Estimates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):405–410, 1997.

[50] J. Zhang, K. A. Johnson, J. T. Malin, and J. W. Smith. Human-Centered Information Visualization. In *International Workshop on Dynamic Visualizations and Learning, Tubingen, Germany*, 2002.