Insider attack detection using netflow records and scan statistics

Schneider, E.

Award date:
2016

Technische Universiteit Eindhoven

Masters Thesis

Insider Attack Detection using NetFlow Records and Scan Statistics

Author:
Erik Schneider

Thesis Committee:
Dr. Jerry den Hartog (TU/e)
Dr. Mykola Pechenizkiy (TU/e)

A thesis submitted in fulfillment of the requirements for the degree of Masters of Computer Science Engineering

in the

Security Group
Department of Mathematics and Computer Science

25 March 2017

Declaration of Authorship

I, Erik Schneider, declare that this thesis titled, 'Insider Attack Detection using NetFlow Records and Scan Statistics' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

TECHNISCHE UNIVERSITEIT EINDHOVEN

Abstract

Department of Mathematics and Computer Science

Masters of Computer Science Engineering

Insider Attack Detection using NetFlow Records and Scan Statistics

by Erik Schneider

The rise of insider attacks that circumvent or avoid traditional perimeter defense applications has created a need for better detection tools. Concurrent with this trend has been an increase in privacy protections that prevent deep inspection of network packets. One solution to these problems is the use of network flows that contain only meta data of the traffic. These flows can be statistically analyzed to detect the presence of anomalous behavior in an internal network but the volume of flows can be large in modern organizations. The proliferation and improvement of open source tools to process large amounts of data over the last decade has provided the means to implement data science approaches to anomaly detection in large production networks.

This paper will investigate a lightweight and scalable algorithm designed to detect malicious insider attack patterns and enumerate the complications arising from implementing the application in a modern production environment. We empirically test the algorithm's assumptions and report the results of a sensitivity analysis of the application.

Acknowledgements

This project would not have been possible without the support and guidance of my supervisors, colleagues, friends and family. I would like thank my supervisor dr. Jerry den Hartog for the continuous support of my thesis study and research. He demonstrated sustained insight, flexibility and understanding for a topic that lay outside his primary research focus. In addition, he was instrumental using his contacts to help me conduct my research at the financial institution. He later recruited Mykola Pechenizkiy to sit on my thesis committee. Mykola provided valuable feedback after I presented my project to him and helped properly structure my thesis and presentation. Thank you Mykola.

I would also like to extend my gratitude to my classmates at TU/e in general and those in the Kerckhoffs Institute program in particular for welcoming me as a mature student among their ranks. I have greatly benefited from your enthusiasm and knowledge.

Finally, I would like to thank my family for their support and encouragement over the years. I'm especially happy that my dying Estonian mother will see her son graduate and enjoy the experience of living in Europe.

Erik Schneider, Amsterdam, February 2016

Contents

List of Figures

List of Tables

Study after study show that greater satisfaction comes from the means, the chase, the journey and the intrinsic.
- Richard L. Daft

Chapter 1

Introduction

1.1 Network Defense

Information has always been of value to adversaries and its protection has been the focus of increasingly sophisticated schemes and ciphers. The complexity of modern systems has increased both the effort level required to obtain sensitive information and the number of attack vectors available to an attacker. These attack vectors have only increased as businesses operate globally, employees work remotely and mobile devices have proliferated.

In the face of these issues, organizations have responded by deploying multiple layers of security controls throughout their information technology systems, a strategy known as defense in depth [3]. Historically, organizations have focused on monitoring traffic that cross network perimeters [4] but this approach has several limitations. First, perimeter defense tools, such as firewalls and forward proxy servers, are not effective against insider attacks. Most are commercial products that are expensive and require considerable experience to configure it properly. Misconfigurations may lead vulnerabilities that hackers can exploit.

Network intrusion detection applications are a common tool in such systems to detect whether outer defensive mechanisms have been breached or evaded. Attackers often bypass these defenses by manipulating people to provide access credentials, known as social engineering, or exploiting vulnerabilities in applications that have not be updated or have yet to be discovered by security community. Once outer defenses have been breached, new applications are required to safeguard sensitive data.

1.1.1 Intrusion detection systems

A common interior defense application is a network intrusion detection system (NIDS), which attempts to discover malicious activity by monitoring network traffic. These systems are generally classified by their approach. Signature-based IDSs inspect network packets and compare them against a database of signatures or attributes from known malicious threats. The advantages of signature-based approaches is that they are fast, they result in very low false positive signals and they provide analysts with important information regarding what type of attack generated the alarm. Their primary disadvantage is that the signatures must be known before detection can occur. Attackers continually evolve new intrusion methods designed to evade signature-based systems and reduce their detection rates. In addition, the expanded use of encryption can reduce or eliminate the effectiveness of signature-based schemes because comparisons against plaintext malware signatures are computationally expensive and may even be impossible [5].

Statistical anomaly-based algorithms attempt to correct this shortcoming by detecting deviations from a baseline of network behavior that is considered normal. When the deviation is large enough, then it is informally considered an outlier [6]. But what is an anomaly formally? Constructing a precise definition can be difficult but we adopt Hawkins' formulation, which is also endorsed by the data scientist at the financial institution (FI) where we conducted our research: an observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism [7]. The focus on the underlying mechanisms generating the data is a crucial concept. For example, a data point that is separated by a far distance from a cluster of other observations may not be anomalous if the underlying process can be expected to generate such extreme observations on occasion. When the observation is not expected, an alert is raised to security analysts that anomalous behavior may be present in the network. Thus if novel forms of attack generate anomalies in network behavior, then statistical-based systems offer a solution to their detection [8].

A real-world complication of using 'normal' mechanisms to define baseline behavior is that these mechanisms must be allowed to evolve or even abruptly change over time. These changes can be the result of new tools or procedures to accomplish the same tasks or represent new responsibilities for the user in question. Thus the baseline for normal behavior must also evolve and be periodically recalibrated to reduce the amount of false positive and false negatives. The optimal frequency of recalibration merits further research but we consider it outside the scope of our project.

1.1.1.1 Insider attacks

Statistical-based network intrusion detection systems can be especially useful in identifying insider threats, including not only those who attack the system from within but also authorized users who violate organizational security policy by escalating their privileges [9]. The following quote from the U.S. National Institute of Standards and Technology (NIST) describes the problem of privilege escalation:

> System controls are not well matched to the average organization's security policy. As a direct result, the typical user is permitted to circumvent that policy on a frequent basis. The administrator is unable to enforce the policy because of the weak access controls, and cannot detect the violation of policy because of weak audit mechanisms. Even if the audit mechanisms are in place, the daunting volume of data produced makes it unlikely that the administrator will detect policy violations.[10]

These insider threats are important to detect because they evade technical perimeter solutions and because they may be accomplished using social engineering, an effective and increasingly popular from of human manipulation. The number of insider incidents has increased geometrically over the years [11] and the FBI and the Department of Homeland Defense have recently warned businesses that security incidents involving malicious insiders is increasing [12]. In addition, PricewaterhouseCoopers found that a third of 500 US businesses they surveyed believe insider incidents are more costly than incidents generated by outsiders. They also found that slightly less than half of the organizations had a plan for insider threats [13].

The power of statistical-based approaches is mitigated by several disadvantages in their implementation. First, statistical approaches often produce a high amount of false positives. Analysts can waste several hours investigating each false positive so reducing their number is a high priority for any real-world system. Second, the alerts generated by statistical approaches may be difficult for an analyst to interpret. Unlike like signature-based approaches that usually give the analyst the name of the malware or attack, statistical applications provide the name(s) of the nodes generating the alert and an anomaly score reflecting a statistical measure of deviation. A third complication of statistical approaches is that they require large amounts of data and/or large computational resources. These requirements reduce their scalability for use in large systems. Finally, several statistical-based detection approaches, such as k-means clustering, view the problem of detection from a top-down perspective, considering the entire network as a whole. Attacks, however, may be very localized in nature and short in duration. For this reason, a bottom-up approach may be better suited to uncover this anomalies [14].

Our review of the literature (see 3.1) indicates that much of the research on statistical anomaly-based detection methods has been tested on simulated data. This is the unfortunate consequence of a lack of publicly available network data containing attacks. While simulated networks may model some aspects of real systems with fidelity, they often do not capture the complex nature of all the mechanisms at work on the network. As a result, researchers cannot state with great confidence that an approach found effective on a simulated network would achieve the same results on a complex modern one.

The above limitations of statistical-based detection systems may explain their low adoption rate among organizations [8]. This is unfortunate because the promise of detecting novel attacks is important. Virus scanners, for example, have lost their effectiveness in catching viruses over the last decade [14, 15]. Ross Anderson states, 'while antivirus software might have detected almost all of the exploits in circulation in the early 2000s, by 2007 the typical product might detect only a third of them.'[14] In addition, the use of zero-day attacks, i.e. novel attacks or threats that provides vendors zero days to develop patches to the discovered vulnerability, has increased in recent years [16]. Both of these trends, as well as others, suggest the efficacy of signature-based approaches may be declining. At a time when more data is being held and transferred in electronic form, this is a serious problem. In response, we investigate a localized, scalable statistical-based detection system in a large, complex network environment.

1.2 Our Approach

The primary focus of our algorithm is to detect attackers that have gained access to the internal network of a large organization. Were agnostic about the means the adversary used to obtain such access, whether it be a malicious insider or some new malware that have evaded all perimeter defenses. The attacker will likely have little choice of computer, also known as a node, she has gained control over on in the network. It is likely that this node fails to contain sensitive information, which is often surrounded by yet more defenses, but control of the node will allow the attacker to search nearby targets and to move across the network, known as traversing, to eventually exfiltrate the desired data.

The presence of an attacker scanning and infecting new nodes represents an additional mechanism at work in generating network traffic. This behavior should manifest itself as increased activity along the connections between two computers. Connections between computers, known as arcs in graph theory, can be measured using network collection and analysis tools, including Ciscos NetFlow [17]. NetFlow groups connections into 'flows

using packet header information, such as source and destination addresses and ports, and provides records describing characteristics of the communication, such as duration, TCP flags and number of bytes sent. By counting the number of flows that occur within a fixed period, one may begin to develop a normal profile for the communication between two specific computers.

Once a normal activity profile has been estimated for an arc, then current activity can be monitored and the likelihood of that activity level can be estimated in reference to the arcs historical behavior. Our research aims to discover if observing arc activity levels can provide reliable detection of traversal attacks.

We believe this approach is important for several reasons. First, it is fast and scalable because it only requires the application to look at packet headers and not inside the packet payload. Full payload inspection increases computation time, reducing scalability, and may be negated by encryption.

A second advantage of our method is that it avoid packet payload inspection, which may infringe on privacy rights. The latter point is especially important in Europe as the European Union is set to enact the General Data Protection Regulation in 2018, which will impose restrictive conditions on data controllers when processing user data or profiling users [18, 19]. Our approach enhances privacy by focusing only on packet header information and by profiling arcs, not users.

A third benefit of our approach is that it utilizes existing tools that are already widely adopted and, in some cases, inexpensive. NetFlow collection is a module on many Cisco routers, the predominant router manufacturer on the planet. Furthermore, the development of Big Data tools has provided an easy means to implement machine learning and other algorithms to analyze these flow records. Open source software projects including Hadoop and Apache Spark provide free software tools to create and execute algorithms. Other open source projects, such as Apache ElasticSearch and Kibana, offer users search and visualization capabilities assess the results of these algorithms. The affordability of these powerful tools allow small and medium organizations to implement sophisticated detection systems within their limited budgets, increasing security for everyone.

As a fourth benefit, our approach is easily interpretable by security analysts and provides information on the likely infected node and the time of infection. We feel this is an improvement over some machine learning algorithms that do provide a good understanding of why the anomaly was created. For our approach, the explanation for an alert is that the activity along a directed path has exhibited an increase in flows that differ greatly from historical norms, possibly indicating that a new, malicious mechanism is operating

along the path. Because the path is directed, the analyst can start his inspection at the first node in the path to see if it exhibits other indications of compromise.

A fifth advantage of using our algorithm is that it is bottom up. We look at anomalies along each individual arc in the network and then widen the view to distinguish common attack shapes. In addition to catching highly-localized and short-lived attacks, the approach can reveal several concurrent attacks within the same time period. Localized disruptions or missing data from one part of the network will also not affect the algorithms detection effectiveness in the remaining areas of the network. Finally, we presume no specific 'attack pattern' but instead allow many patterns that involve traversals and scanning to be detected.

To ensure the applicability of our results, we tested our algorithm by implementing it on a complex, real-world network at a large international financial institution (FI). Although information systems can differ greatly according to their environment, we believe testing algorithms on real data is the best method for confirming theoretical conjectures and will encourage organizations to implement similar approaches.

1.3 Thesis Overview

The remaining thesis is structured as follows:

Chapter 2 provides a detailed look at the problem statement, our objectives and the research statements under investigation.

Chapter 3 reviews the most recent and relevant academic literature related to network anomaly detection systems.

Chapter 4 describes the theoretical basis and the statistical models used in our behavior-based approach.

Chapter 5 details the data and tools used as well as the challenges encountered during the execution phase of our project.

Chapter 6 highlights the results achieved from our implementation.

Chapter 7 interprets our results and provides insight into the costs, benefits and alternatives to localized behavior-based anomaly detection algorithms. In addition, areas for future research are suggested.

Chapter 8 concludes this thesis with a review of our research questions and a summary of our answers to them.

Chapter 2

Research Methodology

This chapter highlights the knowledge gap this project addresses and defines the problem formally. To further provide clarity, we identify and provide a motivating example to illustrate the high-level objective of our research. We then articulate the relevant research questions, their relevance and our approach to answering them.

2.1 Knowledge Gap and Problem Statement

A fundamental problem for academics researching anomaly detection systems is the lack of available data sets that contain real traffic and up-to-date attacks. Many researchers have tested the efficacy of their algorithms using the KDD Cup 1999 data set but there exist several problematic factors with its use and the results obtained on it may not be applicable for contemporary real-time business settings.

The data set was the basis of a competition during KDD-99, the fifth conference in the 'Knowledge Discovery in Databases' series, and is a version of the 1998 DARPA Intrusion Detection Evaluation Program that was developed by MIT Lincoln Labs. It is composed of training and test data and contains a wide variety of intrusion attacks within a simulated military network. The set is attractive because the test data does not conform to the same probability distribution as the training data and contains attack types not found in the training data [20]. These attacks, however, have not been updated during the 17 intervening years and likely do not reflect current malware and intrusion techniques. Furthermore, the data fails to provide source/destination addresses or time stamps, thus limiting the type of algorithms that may be tested on the set. Indeed, researchers hoping to obtain favorable comparisons with other algorithms in the literature may favor algorithms that can accommodate the features contained in the

data and shy away from algorithms that cannot. Taken together, these reasons raise questions about the reliability and usefulness of a significant set of anomaly detection approaches in the literature.

Outside of the KDD Cup data set, many researchers will simulate their own data to run their algorithms on. While great care may be taken to ensure fidelity to real data, it can be difficult to model all the variables that generate the traffic patterns found on real systems. In addition, attackers are constantly changing and improving their attack methods. Zero-day exploits are ones that provide developers no prior warning to issue a patch that could close the vulnerability found in their code and have become a popular method of attack because of their power. The behavior patterns generated by these new exploits can be hard or impossible to understand beforehand given their unknown nature. Some researchers, such as Liran Tancman [21], have argued that viruses are evolutionary, not revolutionary, because attackers often reuse code from prior attacks. If true, this suggests that simulated networks and attacks based on historical malware may reasonably represent future malware but we contend that even the same code may produce different side-effects on different (updated) systems. The more variation in the malware code and the attacked systems, the less reliable simulated systems become over time.

The gap in reliability between simulated and real data may explain why businesses and institutions have preferred signature-based schemes to secure their networks despite heavy research in anomaly detection since it was first introduced in 1987. We argue that researchers must work to close the plausibility gap between algorithms tested on simulated data and their effectiveness on real-world traffic before the promise of behavior-based detection systems can be fulfilled. Differences in real-world systems will constrain the applicability of any statements obtained from testing on one system but this limitation applies to nearly all detection results.

Problem statements: Organizations have been reluctant to adopt behavior-based approaches and their potential to detect new forms of attacks, because the lack of real network data prevents the evaluation of detection algorithms in real organizational environments. The lack of data also obscures the complications that arise when algorithms tested in simulated networks are implemented in production systems.

2.2 Objective

The main objective of our project is to provide the research community with our experience implementing a promising, subgraph-based detection algorithm to detect malicious internal behavior on the real data of a large bank. We highlight the complications we encountered and our solutions to overcome them.

Our investigation also considers two sub-objectives. First, we examine the statistical properties of arc activity on real networks. An important consideration for localized, arc based approaches is the independence of arc activity between adjacent arcs because this will impact the appropriateness of certain approaches and their scalability. Second, we examine the sensitivity of one particular anomaly detection algorithm to increases in arc activity. The complexity and variability of modern networks presents considerable challenges to the appropriateness of simple activity-based approaches and we aim to make definitive statements regarding how elevated attacker behavior must be before detection is possible.

2.3 Main research question:

Main research question: How effective is a flow-based anomaly detection algorithm using summary statistics in a large production environment?

Additional research questions:

RQ 1 - What is the state of the art according to the research literature on network anomaly detection?

We review the relevant literature on anomaly detection to highlight the numerous approaches to behavior-based defense systems. The strength and weaknesses of these approaches will lead us to the selection of our chosen algorithm for implementation. The survey will provide context for why we believe the algorithm represents the state-of-the-art in anomaly detection.

RQ 2 - What is the empirical evidence for the distribution of arc activity as measured by flow counts?

An understanding of arc activity is a necessary prerequisite to implementing a detection algorithm based on such activity. An incorrect distribution will degrade the effectiveness of the algorithm and may lead to lower detection rates and higher false positive/negative rates. We empirically test numerous distribution to determine the best fit for arc flow counts in both office building and data center networks.

RQ 3 - What is the empirical evidence for the independence of windowed arc flow counts?

The correlation of flow counts among adjacent arcs has consequences for the anomaly shape scores generated by the algorithm. We test the empirical evidence for arc activity independence in the office building traffic and examine whether the day of the week results in noticeable differences in these correlations.

RQ 4 - How sensitive is the algorithm to increases in flow counts along the core path and adjacent outarcs from the core path?

Malicious insider behavior will generate variable increases in arc activity depending on the nature and sophistication of the attack. We explore several levels of elevated activity along arcs to observe how they affect anomaly scores. In particular we test how elevated activity along individual arcs combine to affect anomaly scores of subgraph 'shapes', which model attack patterns. We also examine whether differences in sensitivity are affected by the general level of network activity by testing light and active periods of the business day. If there are differences, then they may inform practitioners and researchers regarding algorithm selection and modification in specific settings.

2.4 Relevance

Scientific relevance. Our research improves the research community's understanding of the appropriateness and effectiveness of a localized anomaly detection scheme within a complex, real-time network environment. In addition, we provide insight on the algorithmic limitations of open source big data tools. Finally, we outline the challenges of effectively detecting malicious insider in a scalable way that also preserves privacy.

Organizational relevance. Organizations need to continually evolve to remain competitive in the international marketplace while also protecting their core assets. Banks in particular face financial and strategic challenges from ever more sophisticated security threats, from new financial technologies [22] and from the current very low interest rate environment. Their very nature as holders of money, often in excess of one trillion euros, make banks high-value targets for advanced hackers so understanding the effectiveness of proposed detection algorithms and the challenges in their implementation is important. We hope our research informs managers of the promise and limits of detecting insider threats using cost-effective open source tools. In addition, our research will stress the importance of maintaining thorough data catalogs so security and data science can be effective. Finally, as is the case for researchers, this paper will highlight the trade offs between preserving privacy and achieving scalable detection capabilities.

2.5 Research Approach

We implement a localized behavior-based anomaly detection scheme to answer the proposed research questions. In addition, we review and discuss the literature on graph-based anomaly detection approaches to provide context for our implementation and to demonstrate why our model should be considered the state of the art.

The setting for our research is a financial institution (FI) located in the Netherlands. The bank has collected records containing basic network traffic information over the past two years using NetFlow and has made 2015 traffic available for our research efforts. The data contains traffic from two sites. January and February traffic were collected from a medium-sized office building while all later traffic comes from a large data center.

The bank also provided access to a distributed computing environment using many leading-edge open source tools, including Hadoop and Apache projects. Finally, the company's domain and development experts made themselves available for questions and comments on our approach.

Chapter 3

Background

In this chapter we survey the academic literature relating to anomaly detection with special attention to graph-based approaches. We will argue that the lightweight approach by Neil represents the cutting edge in the detection of insider attacks.

3.1 Literature Review

Denning proposed one of the first intrusion detecting systems in 1987. It monitored audit records to flag a wide range of network intrusions and distinguished between host-based and network-based systems [23]. Host-based systems, also referred to as system call intrusion detection systems, examine operating system call traces to identify malicious programs, unauthorized behavior and policy violations. Alternatively, network-based systems attempt to detect intrusions by modeling the data in a sequential manner to discover anomalous patterns (point anomalies) [5].

Since that time, the number of anomaly detection approaches introduced in the literature has vastly increased and there have been several important surveys classifying work in this area. We know highlight several of the most recent and important ones.

3.1.1 Prior anomaly detection surveys

Chandola et al. provided some useful classifiers that highlight how detection methods differ [5], which are summarized in figure 3.1. In addition, the authors specified a unique assumption underlying the notion of normal and anomalous data for every detection technique, which can be used to test the effectiveness of the technique in the domain in question. They also outlined the challenges to anomaly detection, namely that it can be

difficult to distinguish between normal and anomalous behavior, that attackers adapt their behavior to appear normal, that normal behavior is likely to evolve through time and that non-malicious noise in the data makes identifying strong anomalous signals difficult. In the realm of intrusion detection, they noted that unsupervised and semi-supervised anomaly detection techniques are preferred to handle very large amounts of data (often streamed) and to reduce false positive rates.

Nature of Input Data	Anomaly Type	Data Labels	Output
Attributes (a.k.a. variable, feature, field, dimension, etc.) - can be only one or many	Point - instances are anomalous with respect to the rest of the data.	Supervised - training data set has labeled instances for normal and anomalous classes	Scores - a numerical measure of the rareness of an observed instance
Type - binary, categorical or continuous	Contextual - instances are anomalous with respect to a specific context (but not otherwise). Instances are defined according to contextual (related to location or neighbors) and behavioral attributes (all other attributes).	Semi-supervised - training set has labeled instances for the normal class only	Labels - a category label is assign to each instance
Relationships among data instances - sequential (e.g. temporal), spatial, spatial-temporal or graph data	Collective - collections of related data instances are anomalous with respect to the entire data set	Unsupervised - training set contains no labels or no training is required	

FIGURE 3.1: Aspects of an Anomaly Detection Problem

Other well-researched surveys of anomaly detection have appeared in the last two years. Bhuyan et al. [24] described the latest methods, systems and tools for network anomaly detection. They categorized the attacks that are normally encountered in networks and the underlying computational techniques used, which is heavily focused on machine learning. Of particular interest is their section on data sets that are available to researchers. Here they noted the prominence of the KDD Cup 99 database discussed in section 2.1 but also list several other benchmarks. Of the six listed, four consist of simulated traffic, one is specific to distributed denial-of-service (DDoS) attacks and one contains four days of core router level traffic, reducing the data available to train a model. Three real-life data sets are mentioned but attacks in the data sets are either unknown (UNIBS) or simulated (ISCX-UNB, TUIDS). The TUIDS data set [25] appeared promising for our research because it contained flow data but our efforts to obtain a copy were unsuccessful.

Akoglu et al. [1] focused their survey specifically on graph-based anomaly detection approaches and provided the taxonomy shown in figure 3.2. In reference to their taxonomy, our approach is focused on quantitative detection using windows on unattributed (i.e. plain) data in dynamic (i.e. time-evolving) graphs. Window-based approaches are ones that use a number of prior instances to model normal behavior within a certain time period. The authors noted two main challenges associated with tracking large communications networks: (1) the difficulty of monitoring each node individually and updating their correlations and (2) the difficulty of monitoring the large number of arcs in tandem within a highly dynamic network. We note several researchers below, however, that had published approaches to address and mitigate the second challenge listed, including ones that the authors themselves had referenced, which is surprising.

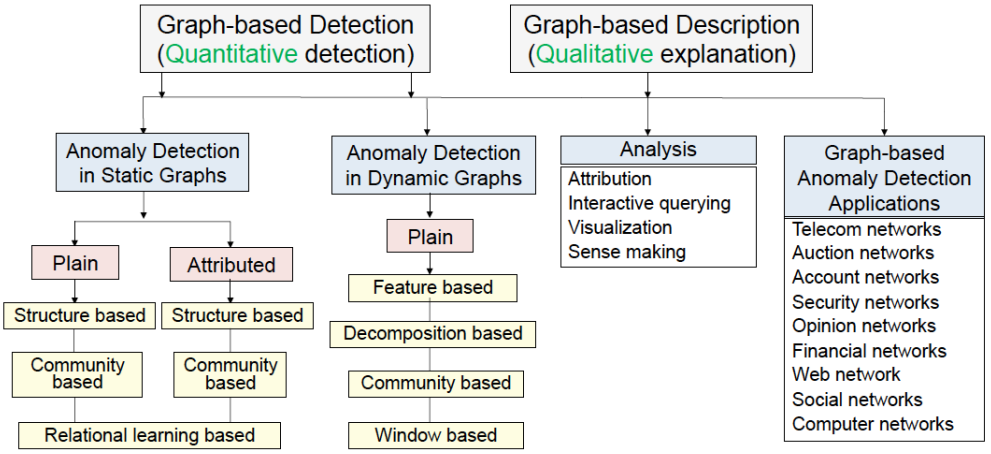


FIGURE 3.2: A Graph-Based Anomaly Taxonomy [1]

Ranshous et al. [26] focused their anomaly detection survey on dynamic networks, which are ones that change their structure through time. Their survey updates the 2006 work of Bilgan and Yener [27] and incorporates more recent research in the field of time-evolving graphs. The authors introduce a classification for different types of anomalies that appear in such environments: anomalous vertices, anomalous arcs (or arcs), anomalous subgraphs and event detection. Within the anomalous subgraphs groups, the focus of our research, the authors describe shrunken communities, wherein a single community loses a significant number of its nodes between time steps, and split communities, wherein a single community divides into two or more distinct communities between time steps, as two specific subtypes. Note that these two events occur in dynamic networks that evolve over time and not in a single graph. The authors only consider machine learning techniques to generate communities (i.e. subgraphs) and propose the two-tier system shown in figure 3.3 to classify anomaly detection methods. This classification differs from Akoglu's version for dynamic graphs, which was based on how the graph was summarized and the type of events the algorithms were designed to detect (feature-based, decomposition-based, community-based or window-based).

Sarma and Sensarma [28] is the most recent survey of graph-based anomaly detection methods. The authors pay special attention to graph-based approaches to fight cyber crime and provide examples and results for several applications in the literature. Their classification of detection methods, however, is almost identical to that of Akoglu and their directions for future work is far less complete.

A review of these surveys highlights the trend toward using machine learning algorithms to detect anomalies. This trend rides on top of other trends, including the rise of open source distributed computing and the expanded collection of network behavior information. Our approach does the same.

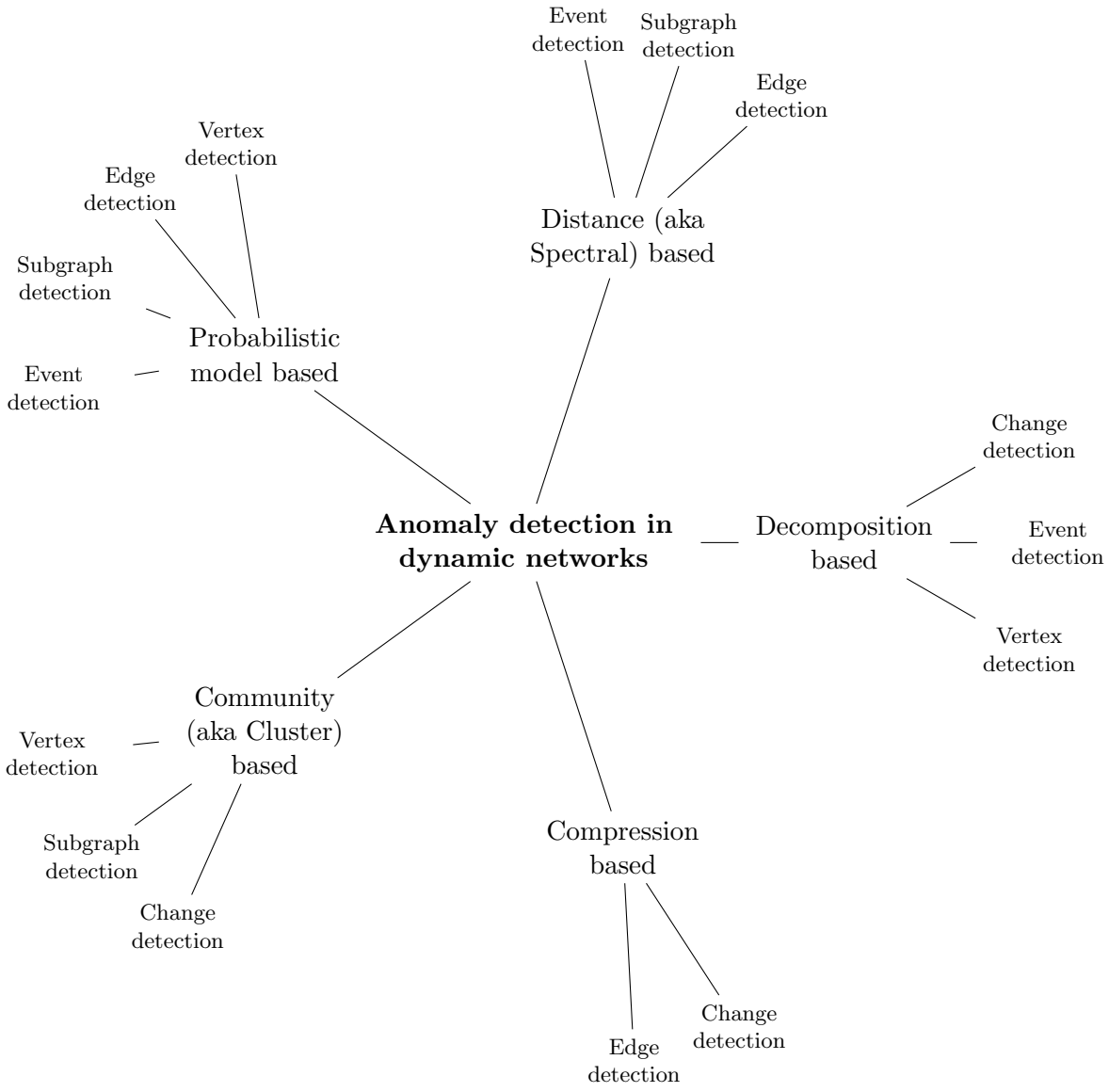


FIGURE 3.3: A Classification of Network Anomaly Detection Methods

3.1.2 Time-dependent subgraph statistical approaches to anomaly detection

We now explore recent work in the specific areas of statistical-based and graph-based approaches to anomaly detection. These approaches typically start by examining user behavior to generate 'normal' profiles that later profiles can be compared against to determine if an anomalous event is occurring. These profiles may include measures of a user's activity level or the distribution of activity over categories, among other examples. New profiles are continually generated by period, and the statistical difference between the current profile and an historical one that is considered normal is calculated to generate an anomaly score. An advantage of the statistical approaches is that they don't require prior knowledge of attack behavior, which could bias the approach to known

attacks. Their disadvantages include the difficult problems of balancing false positives and false negatives, ensuring that malicious behavior is not included in the training data and finding accurate and stable statistical representations of user behavior, which may not be possible [8].

Researchers have explored how best to describe large graphs using common graph structures (e.g. stars, cliques, etc.). Koutra et al. [29], for example, developed a lossless compression algorithm based on the Minimum Description Length (MDL) principle that is parameter free. Mongiovi et al. [30] tackle the problem of finding anomalous subregions by relating it to the NP-hard problem of finding the Heaviest Dynamic Subgraph. Their algorithm alternates between detecting subgraphs that maximize anomaly scores for a given interval (spatial), and detecting time intervals that maximize scores for a given subgraph (temporal). Although 'interesting' sub-structures discovered by these methods may identify intrusion attacks, they are not designed for that purpose and provide little actionable information for analysts.

Heard et al. [31] applied a two-stage Bayesian approach to detect anomalous subregions in social networks. An anomaly score was calculated for every arc by counting the number of interactions between each pair of nodes and then p-values were estimated based on Bayesian learning of the count distributions. Local anomalous regions were then detected by applying clustering techniques. The algorithm used sequential analysis to detect changes in new graphs and used retrospective analysis to update the baseline profile for the arc. Aggarwal et al. applied a similar approach to graph streams [32] but differed by calculating arc probabilities between pairs of network partitions in the incoming graph stream.

Along the same lines, Priebe et al. [33] introduced the use of scan statistics on arc values to detect anomalous subgraphs in a time series of Enron email graphs. Scan statistics, also known as 'moving window analysis', calculate a local statistic (e.g. the count of a specific type of event) for small 'windows' over the data and tests the maximum of these locality statistics, known as the scan statistic, against a critical value to determine an anomaly score for the window. The window of analysis may be a disjoint subregion in the case of static graphs or a disjoint time segment in the case of a dynamic graph. Priebe defined subgraphs to represent be kth-ordered neighborhoods around every node within the data (i.e. the set of all nodes that could be reached by traversing k adjacent arcs starting from the origin node). By varying the value of k, investigators can adjust the granularity of their analysis to subgraphs of differing complexity.

The above subgraph approaches are attractive to network attacks because of their view on localized areas of the network, and attacks can be very localized. Priebe use of scan statistics introduced a lightweight and efficient method for identifying 'interesting'

subgraph, which helps the scalability of the algorithm to large data sets. Despite this appeal, the benefit of scan statistics to detect network attacks was unknown until the work of Neil. We know turn our attention to his research.

3.1.2.1 Neil's approach for the online detection of locally anomalous subgraphs

Neil dissertation [34] focused on used summary statistics over subgraphs to detect network traversal attacks. His approach, which is the foundation of our approach, is similar to Heard et al. [31], Aggarwal et al. [32] and Priebe et al. Priebe et al. [33] in that he aggregated local statistics to form subgraphs for use in anomaly detection. Whereas the previous authors focused on social network graphs, however, Neil applied the technique to the network intrusion domain. In addition, his use of window-based events differs from Heard's and Aggarwal's stream-based approach.

Neil does not require the subgraphs to be disjoint within a time window. Rather, the local statistic is calculated on the most granular level, a directional connection between two computers, which is then aggregated to form the subgraphs. These connections, called arcs in graph theory literature, likely exhibit a pattern of behavior through time that reflects the normal behavior of the mechanism generating the data, such as an employee performing daily tasks on the machine.

A testament to Neil's approach is that it was to be incorporated into Los Alamos National Laboratory's next generation anomaly detection system. Los Alamos conducts top secret military research and protecting it against attacks is a critical priority. In addition, the laboratory's system generates terabytes of data daily and thus requires scalable applications by design. That a scan statistic approach to detection was selected to secure this network signals its power and efficiency. Its use of only packet header information is important as expanded use of encryption and privacy concerns have limited the ability of detection systems to observe packet payload information.

Neil has since applied for a patent for the scan statistic approach outlined in his dissertation [34] and one using new arcs as a anomaly measure [4], an enhancement suggested in section 7.1.2 of his thesis. The new arcs approach uses an asymmetric exponentially weighted moving average (AEWMA) algorithm, which updates the probability of a node generating or receiving an arc unseen in the training set. The assumption underlying the approach is that new arc creation is a natural process in a dynamic network and that individual nodes have a unique rate at which these new arc appear incident to them. Furthermore, AEWMA handles the reappearance of arcs that have not been seen for an extended period of time and are considered 'new' again. The algorithm, which had

first been applied in a security setting by Tang et al. in 2014 to detect Low-rate Denial of Service attacks [35]. Previously, Pincombe had applied an autoregressive moving average approach to distance measures [36].

We do not believe our application violates his patents because (1) his patent applies to embodiments where the logic is implemented in hardware modules and (2) our application makes significant alternations, we would argue improvements, to his algorithm (see section 5.2).

3.1.3 Statistical descriptions of arc activity

Researchers have investigated several characteristics of arc behavior, including the distributions of byte sizes and connection end times [37]. Wright et al. [38] assume that the number of connections in a time window follows a Gaussian distribution in their attempt to infer application protocol from TCP traffic by only observing packet size, duration and direction. Benson et al. [39] observe lognormal distributions for data center SNMP traffic and describe a method for estimating lognormal parameters. Garsva et al. [40] use the Kolmogorov-Smirnov test to determine goodness-of-fit for packet inter-arrival times and found that the Pareto Second Kind distribution provided the best fit for the majority of their experimental data.

Lambert and Lui [41] evaluated the negative binomial, Poisson and lognormal distributions as possible models for hourly network counts of packets and provide evidence that the negative binomial distribution results in the most uniform distribution of p-values when parameters are exponentially updated. The use of p-values adjust for the lack of stationarity from one count of flows to the next. This result suggests that the model captures day and week periodicity, long-term trends, long tails, and regional differences in the flows. The authors note that thresholds should adapt automatically to patterns without looking at historical data to account for the variability and, at times, lost data. Practitioners require an accurate distribution model, especially in the tail regions, to set appropriate thresholds.

To our knowledge, no researcher has presented specific findings on how flow counts are distributed in set intervals, however. The experts we spoke with at the national applied science institute also had no information about these distributions, but expressed interest in them. Our work contributes to this gap by testing the fit of many distributions to observed arcs.

3.2 Literature Discussion

Three important but often neglected topics in many of the surveys and paper we reviewed are privacy protection, localization and ease of anomaly score interpretation. Many behavior-based techniques create 'profiles' of users, which the EU General Data Protection Regulation only allows under restrictive conditions. Detecting small events at the local level matters because traversal attacks may be only involve a few nodes that already communicate frequently. The interpretation of anomaly scores by security analysts is critical if an algorithm is to be included in their regular workflow. What may be interpretable to a data scientist may not be interpretable to a security analyst so an anomaly score should have a straightforward cause in the analyst's head if the algorithm hopes to be widely adopted by organizations. We believe our algorithm addresses all of the above considerations while also providing detection of anomalous network traversals. We provide more details of our model in the following chapter.

Chapter 4

Attacker Model and Detection System

In this chapter we introduce the attacker model of interest and show how it informs our detection method, which closely follows the one proposed by Neil [4, 34] for detecting anomalous subgraphs using summary statistics over NetFlow records. We begin in section 4.1 by describing a specific attack model and what areas within a corporation it affects. This will complement the discussion found in 1.1.1.1. In the section 4.2 we then explain our network model and show how a computer network can be represented as a graph. Subgraphs, and specifically the two subgraphs relevant to our attacker model, are considered in section 4.3. We discuss the potential power of new arcs as an attack indicator in section 4.4 before describing the statistical models we use to generate anomaly scores in section 4.5. Finally, a summary section is provided to briefly review the important concepts in our models and our approach.

4.1 Attack Model

To clarify the objective of our approach, we describe the an intrusion attack in a generic way. Figure 4.1 [42] depicts the steps in a kill (a.k.a. attack) chain, which is a systematic process to target and engage an adversary to create desired effects [43]. While other kill chain representations exist, some with more stages or different objectives (e.g. denial of service), we have selected this one because of its simplicity in depicting the steps required in an intrusion attack. We note that an insider attack may only involve the final 'Action' stage of the kill chain displayed.

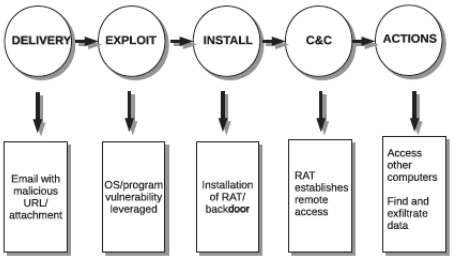


FIGURE 4.1: Attack Chain Model

An asset is considered secure if no adversary of a specified power can achieve a specific break [44]. With this definition in mind, we now describe the attack model our approach is designed to detect. Our attacker is limited in power, e.g. she does not know the normal behavior of the targeted network, and has a specific goal - to traverse the network and exfiltrate sensitive data.

Prior to an attack, an attacker will often research, identify and select targets based on desired objectives in a stage called Reconnaissance (not depicted). This goal of this stage is to collect mailing lists and information on the victim's system(s). Then in the Weaponization stage (not depicted) a remote access trojan (RAT) is coupled with an exploit to take advantage of a likely vulnerability in the target system. In the Delivery stage, the adversary attempts to transmit the malicious payload by via an email attachment, a website, a removable media device (e.g. a USB memory stick) or other method. The Exploitation and Installation phases occur when the intruder's code is executed on a victim's machine and the RAT or other backdoor is installed.

It's important to note that the attacker often cannot select the machine compromised. One common attack vector is a phishing email, which mimics a legitimate email but contains links to a website controlled by the attacker. When an email recipient clicks the link, the malware is delivered to the user's machine, but the user in question is often unknown beforehand. Advanced phishing attacks attempt to improve initial access by targeting high-value employees with elevated privileges.

Once the malware has been installed, the compromised hosts may beacon out to a command and control server to alert the attacker that it has infected a machine and to await further instructions. As mentioned above, it is unlikely the compromised machine contains the sensitive information desired so the attacker must traverse the network to where that information rests. This stage, called the Action stage, is the focus of our research.

An attacker only needs one way into a system while defenders much monitor and secure all possible entry points. Despite the development of sophisticated perimeter defenses,

hackers continually find ways to evade them, as is evidenced by the perpetual announcements of cyber attacks in the press. In addition, disgruntled or compromised insiders control machines already behind these defenses. We are agnostic about the method of infection and care only that a machine is under the influence of a malicious agent. As a traversal occurs in the final stage in the attack chain, catching traversals represents the last, or nearly last, line of defense. Because the malicious traversal represents an additional mechanism at work on the arcs, we only consider elevated arc activity as a possible indicator that malicious activity is present. Anomalously low arc activity is assumed to be benign.

One disadvantage for an outside attacker is that she does not have intimate knowledge of the normal behavior of the network. The adversary or malware must continually discover network neighbors as they progress through the network, generating anomalous records in NetFlow and audit logs. Our goal is to use summary statistics on NetFlow records to catch this anomalous behavior.

4.2 Graphs and Windows

A graph is an abstract data type that consists of a set of nodes (a.k.a. vertices) and the connections between these nodes. These connections are called 'edges' when the direction of the connection is unimportant or not applicable and are called 'arcs' when the direction matters. Graphs are often used to represent communication networks because they are designed to model pairwise relations between objects. In the case of an IT network, nodes represent computers and an arc represents a directed connection between two nodes. We focus on directed connections, called directed arcs, because we are interesting in identifying the source node of the traversal, if possible [45].

The graph of a complex information system is constantly evolving as new machine join the network while others leave. We look at the network graphs created during specified periods, known as a windows, to reduce the amount of data need to be analyzed and to allow different views of the network to be analyzed. As noted by Priebe et al. [33], a moving window approach is appropriate if the data is assumed to exhibit short-time stationarity properties under the null hypothesis that contiguous windows are homogeneous in some characteristics, flow counts in our case.

The duration of a malicious traversal varies and no one window size is appropriate for capturing all attacks. We investigate the use of several size windows to capture a variety of traversal periods. In addition, we overlap consecutive windows to capture attacks that would otherwise develop across mutually exclusive time periods. Figure 4.2 provides a

visual representation of this windowing process. The time unit, t, is not explicitly defined to reflect the flexible nature of its value and the overlap is set to one third the total size of the window. Attacks may still span more than one window but increasing the overlap size increases the amount of computation and these attacks may still be detecting in the larger window executions of the algorithms.

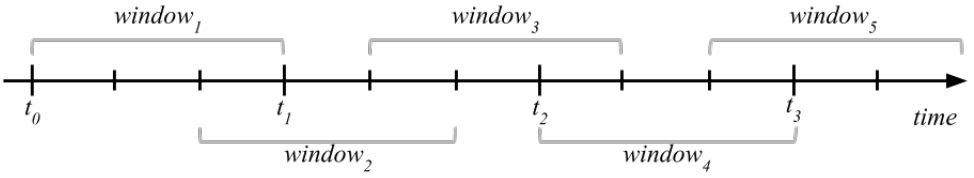


FIGURE 4.2: The windowing process for a time interval t

4.3 Subgraph Shapes

Many approaches to network anomaly detection examine the normal behavior of the entire graph [CITE]. Attacks, however, can be very localized. As a consequence, we focus on examining particular regions, called subgraphs [45], that can be indicative of malicious behavior.

4.3.1 Stars

Stars are shapes created by a single node connecting to immediate neighbors. They are called outstars when all arcs have the same source address and are called instars when all arcs have the same destination address. In an attack scenario, an outstar might be the result of the malware scanning all nearby machine. While unsophisticated and conspicuous, scanning can be effective at gathering information when the network configuration is unknown. Instars, on the other hand, could be any node whose services are queried by other users. Nearly all nodes concurrently handle inbound connections and create outbound ones but separating the traffic by direction allows us to better detect scanning behavior. Although simple in construction, several researchers found them valuable for detecting anomalous behavior [33, 34].

Figure 4.3 depicts an infected node that has generated several anomalous arcs in its search for sensitive data. We discuss when an outward arc is deemed anomalous below but note that only a few rare arcs need be present to make the entire star shape anomalous.

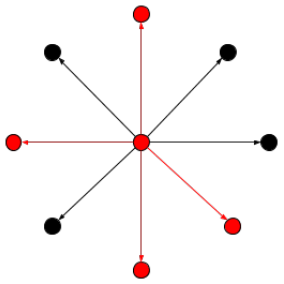


FIGURE 4.3: A star with several anomalous arcs (in red)

4.3.2 Paths

A path is an alternating sequence of nodes and arcs in which all the nodes in the path are distinct [45]. An attacker traversing a network from an initial node to one holding sensitive data will necessarily generate a path of some length, k, except in the unlikely case where the malware or corrupted insider controls the machine holding the sensitive data from the very start.

The process of a traversal attack is shown in figure 4.4. In the first step, the intruder scans nearby machines from a compromised node, creating a star. He then selects one or more of these discovered machines to infect. After that machine is infected, the process repeats - scans, selection, infection - until a machine containing sensitive data is found. The attacker then must exfiltrate the data either directly or via some node in the established path. If a command and control server is utilized and traffic to it has not been detected as malicious by perimeter defenses, then exfiltration through that channel might provide a means to export the data.

The full shape called a directed k-path, or caterpillar, and Neil notes that they have been observed in actual network attacks at the Los Alamos National Laboratory, which researches and safeguards many of the United States military secrets [34].

4.4 New arcs

New arcs that have not been seen historically may be potent signals of malicious activity. The reason, as explained earlier, is that an attacker does not know the historical connection behavior of a node. She may attempt to learn these behaviors by examining the infected node's audit logs but this is only likely if she has gained access to an administrators account. In addition, she may have limited time to execute a full attack because new command and control websites are discovered and entered into threat intelligence databases. Once in the database, system administrators can then update blacklists that

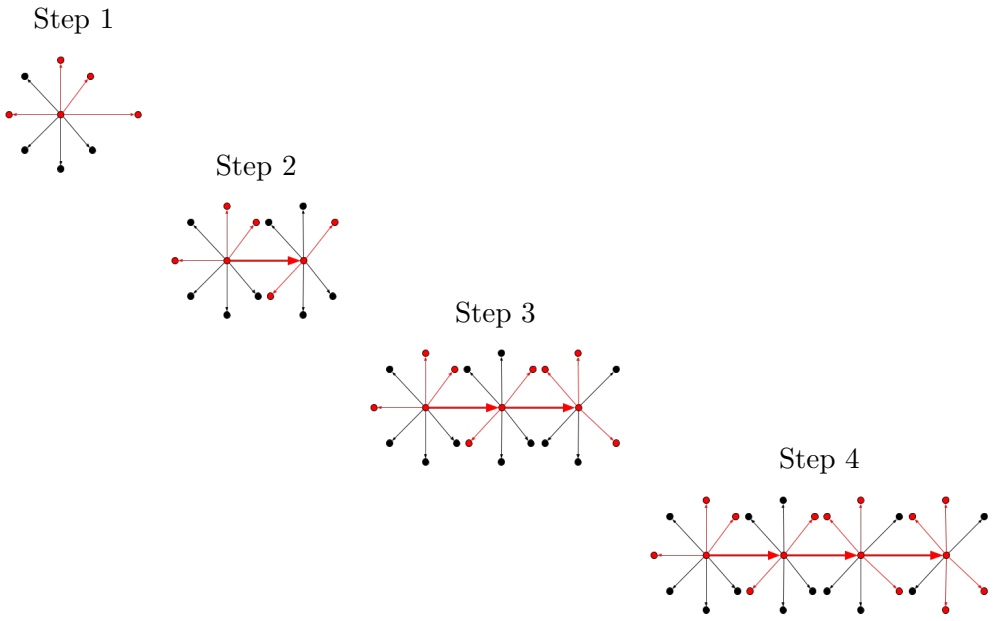


FIGURE 4.4: An infected path consisting of a 3-path core and numerous anomalous outward arcs (in red)

prevent communication to that website, and thus causing the attack to lose control of the node.

New arcs need not be malicious, however. Nodes continually generate and receive new connections as part of their normal behavior. Indeed, some servers, such as Dynamic Host Configuration Protocol (DHCP) servers that assign IP addresses to machines joining a network. Each node will have a rate of new outward arcs and inward arcs that fluctuates through time. Our implementation identifies the nodes generating the greatest amount of new outarcs per window as a proxy for naive attacker behavior. A more sophisticated approach is discussed in section 7.1.2, but it requires the system to maintain a large amount of state, which goes against the lightweight and scalability goals of our application

4.5 Scan Statistics and Anomaly Scores

4.5.1 Stars and paths

Network traffic can be analyzed according to different measures, such as connection appearance times or connection duration. We chose to focus on arc activity because it can capture to new mechanism, such as malware, operating on a given arc on top of normal use. A second reason is that it minimizes preconceived notions about how attacks should evolve, allowing novel attacks to be captured. For example, arc activity

makes no prior assumptions about what TCP flags will be set. While certain TCP settings have been associated with known attacks, they can be considered 'signatures' much like Common Vulnerabilities and Exposures (CVE) Identifies prejudice signature-based NIDS to capturing only historical attacks. Our goal in part is to capture new attacks and we consider the least informative model the best approach to accomplish this. Finally, arc activity can be measured using NetFlow flow counts, described in 6.1.1.1, and calculated quickly using Big Data tools, improving the scalability of the algorithm.

Each arc will exhibit arc activity that is likely to be unique over the course of time. We believe that the mechanisms generating observed flow counts can be modeled according to common probability distributions.

4.5.1.1 Gamma and normal distributions

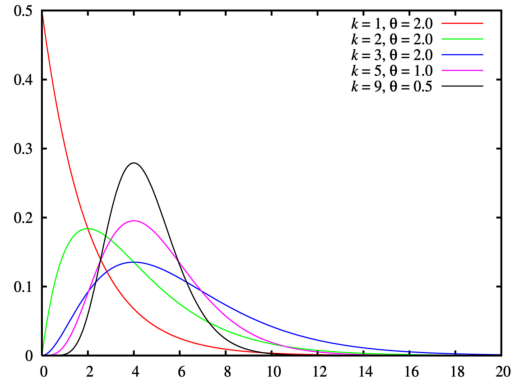
Selecting the correct distribution to model arc activity behavior is an important design choice for our approach. An poor choice that does not model the underlying mechanism(s) well, especially in the tail region, may lead to an excess of false positives, if the kurtosis is lower than it should be, or false negatives, if the kurtosis is higher than it should be.

There are other, practical considerations that may influence the choice of distribution, however. Neil, for example, chooses the gamma distribution in part because of its ability to assume different shapes [34]. In fact, several well known distributions, including the exponential an chi-square distributions, are special case of the gamma. This property allows a likelihood ratio test, detailed in section 4.5.1.2, to be performed using a gamma.

The gamma distribution (figure 4.5) models systems that generate positive values and it is the maximum entropy probability distribution, i.e. for a random variable that has fixed mean and unknown variance. This property is attractive because maximizing entropy achieves the Principle of Minimum Bias, reducing the effect of prior information in selecting a restrictive distribution [46]. A competing choice is the normal distribution (figure 4.6), which is also maximum entropy but for a specified mean and variance. We implemented our algorithm to use both the gamma and the normal distribution for exploratory purposes.

An another attractive distribution we considered is the Poisson. A Poisson process in one where events occur continuously and independently at a constant average rate and the probability of a given number of events occurring in a fixed interval of time is given by the distribution of the same name. A unique property of the Poisson distribution is

Gamma pdfs for different parameters [48]



$$\text{pdf} = \frac{1}{\Gamma(k)\theta^k} x^{k-1} e^{-\frac{x}{\theta}}$$

$$\text{mean} = k\theta$$

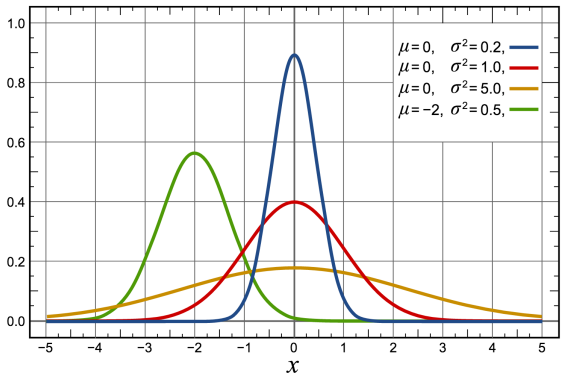
$$\text{var} = k\theta^2$$

$$\text{skewness} = \frac{2}{\sqrt{k}}$$

$$\text{kurtosis} = \frac{6}{k}$$

FIGURE 4.5: Gamma Distribution PDFs and Moments

Normal pdfs for different parameters [49]



$$\text{pdf} = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$\text{mean} = \mu$$

$$\text{var} = \sigma^2$$

FIGURE 4.6: Normal Distribution PDFs and Moments

that its variance equals its mean, limiting the processes that can be modeled by it. We choose not to implement the Poisson as a result of this limitation and believe the normal distribution, with its variance free to adopt any value, can capture Poisson processes reliably. Poisson distributions do exhibit a slightly positive skew and kurtosis that is inversely proportional to its mean but for large values, which is the case for the flow counts on most arcs, these higher moments become negligible.

The gamma and normal distributions differ in their shapes and in their statistical moments. The normal is completely defined by its mean and variance, i.e. its first two moments, and thus exhibits neither skewness nor kurtosis ('fat tails'). It a symmetric distribution about its mean. The gamma, on the other hand, will always show some positive skewness and some positive kurtosis [47]. These higher moments are defined by a shape parameter k and a scale parameter θ.

4.5.1.2 The generalized likelihood ratio test

A generalized likelihood ratio test (GLRT) is a statistical test that allows one to compare the goodness of fit of two competing models, one of which is a special case of the latter. It can used to calculate the probability of the observed flow count for a given arc in the current window based on historic parameters.

$$\begin{aligned}\lambda_\gamma &= -2\ln\left(\frac{\text{likelihood for null model}}{\text{likelihood for alternative model}}\right) \\ &= -2\,log\left(\frac{L(\widehat{\theta}(\gamma)|x(\gamma))}{sup_{\theta\in\boldsymbol{\Theta_A}}\,L(\theta(\gamma)|x(\gamma))}\right)\end{aligned} \tag{4.1}$$

In equation 4.1, $X(\gamma)$ denotes the data in the window given by γ, $\widehat{\theta}$ denotes the historically estimated parameters for an arc and $L(\theta(\gamma)|x(\gamma))$ denotes the likelihood of the stochastic process on γ. The null hypothesis, H_0, is that the observed data could have been produced by historical estimates of the parameters. That is, $H_0 : \theta(\gamma) = \widehat{\theta}$. The alternative hypothesis, H_1, is that the values can be better modeled using a restricted subset, Θ_A, of the overall parameter space, Θ. If the null hypothesis is rejected, we conclude that a new mechanism, possibly a malicious one, has been introduced that accounts for the increase in arc activity observed in the current window.

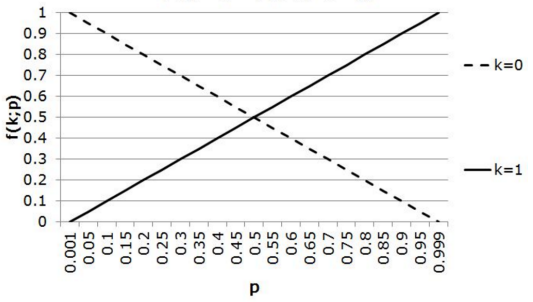
Neil notes that the test statistic, λ, depends on the number of parameters being tested within a window. The solution is to estimate the distribution of λ using p-values, described in section 4.5.1.4. Wilks's theorem states that the generalized likelihood ratio test statistic is asymptotically χ^2 with degrees of freedom equal to the number of free parameters in Θ [50]. This means we can use Fisher's method, described in section 4.5.1.4, to obtain our p-value. However, we need to account for one more aspect of arc activity before a p-value can be estimated.

4.5.1.3 Zero-inflated distributions

Arcs are not active during every window. Even in the FI's data center network, very few arcs were active in every window during a month and many had low numbers of windows with any activity. As a consequence, any distribution of the flow counts will have a large point mass at zero. This is not ideal for estimating distributions.

One solution is to model arcs using zero-inflated distributions. That is, we use a Bernoulli distribution to estimate the probability of any arc activity and another distribution, the gamma and normal in our case, to model the flow counts when there is arc activity.

Probability mass function [51]



$$\text{pmf} = \begin{cases} 1-p & \text{for } k=0 \\ p & \text{for } k=1 \end{cases}$$

$$\text{mean} = p$$

$$\text{var} = p(1-p)$$

$$\text{skewness} = \frac{1-2p}{\sqrt{p(1-p)}}$$

$$\text{kurtosis} = \frac{1-6p(1-p)}{p(1-p)}$$

FIGURE 4.7: Bernoulli Distribution PDFs and Moments

A Bernoulli distribution (figure 4.7) models a discrete random variable that can only take values 0 or 1. The probability of 'success', that is $x = 1$, is p while the probability of 'failure' is $1 - p$.

Let Λ denoted the distribution of our GLRT test statistic λ. We now model Λ as our p-value estimate according to equation 4.2.

$$\Lambda_\gamma = B_\gamma X_\gamma \tag{4.2}$$

where B_γ is the Bernoulli estimate that an arc has any activity in window λ and X_γ is either the Gamma or Normal probability of the observed flow count, if present. The latter probability is estimated using historically estimated parameters for the appropriated model and the value is equal to $1 - CDF_x$, where CDF is the cumulative distribution function.

4.5.1.4 Combining p-values: Fisher's method

Before combining individual arc p-values to calculate an overall probability for an attack shape, we must consider the independence of arc flow counts. We found mild positive correlations among the arcs in the FI network (see results and discussion in section 6.2.2). That said, we assume an independence between these values in our model because, from a practical perspective, the amount of computation and memory required to maintain a variance/covariance matrix for tens of millions of unique directed arcs (e.g. 48 171 388 directed arcs were enumerated in the FI's July network) conflicts with our stated goal of implementing a lightweight and efficient algorithm.

Assuming adjacent arc activity independence has important consequences for resulting anomaly signals. P-values will likely be lower (i.e. indicate a greater anomaly) than in

the case where the arcs have a positive correlation. Conversely, p-values will likely be higher under an independence assumption than they would be when a negative correlation exists between adjacent arc activity. To see why, consider the case of two arcs that exhibit a correlation coefficient of -1. That is, whenever the activity on one arc increases by x percent, the activity on the adjacent arc decreases by x percent in every instance. Our model only treats elevated values of flow counts as anomalous so if higher than normal counts are observed on both arcs, then this would result in a positive correlation, which given their extreme historical negative correlation should result in an even lower p-value. The same reasoning supports the conclusion that p-values will be lower under independence than when a positive correlation exists between arcs. Intuitive logic suggests that there are more positive correlations than negative ones among arcs because the mechanisms generating the former (e.g. a common resource requires traversing several nodes) are easier to conceive but this conjecture should not be assumed as fact. If true, however, then our model may generate more false positives than necessary.

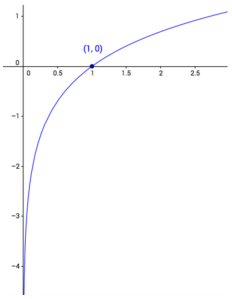
Our assumption of independence allows us to combine p-values using Fisher's method 4.3. This method is attractive because it encapsulates a wide range of values [52] in a small, understandable number. A key insight of the approach is that the sum of log-p-values follows a chi-square distribution with $2k$ degrees of freedom, where k is the number values being combined. Chi-square distributions are commonly used to infer properties of the underlying population or mechanism, which is our goal in analyzing sampled arc activity.

$$X^2_{2k} \sim -2\sum_{i=1}^{k} \ln(p_i) \tag{4.3}$$

To understand how Fisher's method converts p-values to chi-square values, we note that a chi-square distribution can be conceived as the sum of the squares of k independent standard normal random variables. P-values should be uniformly distributed on the interval $[0, 1]$ if the underlying model is correct. Taking the natural log of values in this range will result negative values that follow the line depicted in figure 4.8*a*. Reversing the sign of these negative values results in an exponential distribution while scaling exponential distribution values by 2 results in a chi-squared distribution with 2 degrees of freedom. Finally, summing k independent chi-square values results in a new chi-square distribution with $2k$ degrees of freedom [52]. As noted in section 4.5.1.2, we can use a chi-square distribution as our asymptotic GRLT statistic.

Figure 4.8*b* displays a table relating probabilities to Fisher scores. As one can see, a decrease in a full order of magnitude results in an increase of five points in a Fisher

a. Natural Logarithmic Function



b. Sample Fisher Scores

Probability	FisherScore
10.00000000%	4.61
1.00000000%	9.21
0.10000000%	13.82
0.01000000%	18.42
0.00100000%	23.03
0.00010000%	27.63
0.00001000%	32.24
0.00000100%	36.84
0.00000010%	41.45
0.00000001%	46.05

FIGURE 4.8: Fisher's Method

score. This is easy for analysts to interpret and still allows a highly anomalous arc to dominate many non-anomalous arcs when entire shape scores are calculated.

4.6 Summary

Our research posits a model for insider attacks that informs the development of a localized anomaly detection system. The attacker model is focused on a malicious agent that his gained access to an internal computer but must traverse the network in order to observe and possibly exfiltrate sensitive data. To discover such attacks, we model the network as a directed graph, where each machine is a node and a connection between machines is an arc. With this construction we can observe the activity rate of each arc, measured using NetFlow flow counts (see section 6.1.1.1), within a time-delimited window.

Historical arc activity rates are stored as parameters to either a normal or gamma distribution and are used to define normal behavior for each ordered pair of computers. The probability of observing a flow count at least a great as the one observed along an arc is calculated during each time window. Arcs and their corresponding probabilities are then aggregated into either star or caterpillar shapes and the most anomalous ones are presented to a security analyst for further inspection. The directed nature of the shapes and the timestamps provided by NetFlow provide the analyst with important information to start the forensic process.

In the next section, we discuss the specifics of our implementation and, in particular, what adjustments to Neil's model are required to properly analyze real-world traffic.

Chapter 5

Implementation

In this chapter we discuss the tools used to implement the model outlined in chapter 4 as well as the necessary adjustments to it to handle real-life patterns. In section 5.1 we describe the tools used to store and analyze FI network traffic. Then, in section 5.2, we detail the changes we made for special arc behavior and for improving processing times. Finally, we provide a summary in section 5.3

5.1 Tooling

This section provides information regarding the tools used to store and analyze data that supports the network and attack models articulated in chapter 4. We highlight several open source tools that allow for convenient and efficient storage and processing of large data sets. The components of this setup are well researched and affordable, allowing even small organizations to implement state-of-the-art intrusion detection schemes.

5.1.1 Apache Spark and Hadoop

Our implementation was written in Python for use on Apache Spark [53], an open source cluster computing framework designed to perform calculations on large amounts of data. Working together with Hadoop Yarn, Spark partitions data, organizes transformations using a directed acyclic graph (DAG) and then schedules tasks to be performed by executors on worker nodes [53]. Figure 5.1 depicts the Spark's distributed architecture.

Spark's fundamental programming abstraction is called Resilient Distributed Datasets (RDDs), which is a logical collection of data partitioned across machines. Application programming interfaces (APIs) are provided that allow users to transform RDDs on

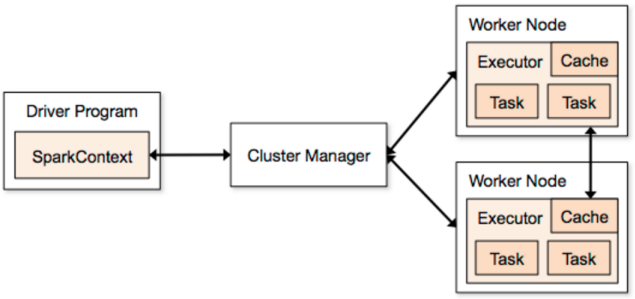


FIGURE 5.1: Apache Spark Work Flow Architecture

a distributed cluster of generic compute machines. A user's main program, called the driver program, uses a SparkContext object to coordinate jobs with a cluster manager, Hadoop Yarn in our case, to efficiently allocate resources and data across the cluster. Executors on cluster nodes are processes that run computations and store data for the application and return data back to the driver when the transformation is complete [54].

Spark applications run on a Java virtual machine (JVM), both in the driver and on the executors, but APIs are available for python, Java, Scala and R languages. We choose to implement our algorithm in python for several reasons. First, the FI's data scientist and contractors possessed considerable knowledge about the language and will be asked to maintain the code after our project ends. Second, python is dynamically typed and less verbose than Java, improving our speed of implementation. Finally, we were experienced using the numpy and scipy libraries for use with scientific computing. Python applications run slower than Java or Scala ones, however, because it needs to translated into Java bytecode. In addition, python library calls must pass from the JVM to the python virtual machine (PVM) and then back to the JVM, adding two extra layers of translation to the compute process.

Spark also has and dataframe APIs for improved processing speeds. Both APIs take advantage of the Spark SQL Catalyst optimizer, which applies logical optimizations, compiles operations into physical plans for execution and generates JVM bytecode. While there is a significant speed improvement using these APIs, the improvements only apply for built-in functions. We take advantage of both the SQL and dataframe APIs when possible but much of our algorithm relies on transformations beyond the capabilities of these built-in functions [53].

We did learn that Spark and/or the resource negotiator, Yarn, does not handle skewed data well. For example, when some nodes have many outarcs, then the (cluster) resource negotiator should partition the data and evenly distribute the computation tasks to executors. A few nodes have a much greater number of outarcs, however, and the working node performing the transformations on such partitions labored immensely

(e.g. 40+ minutes compared to 1-3 seconds for other nodes). When working nodes idle for more than a configured threshold (3 minutes at the FI), they disassociate from the resource manager and make themselves available for other applications, slowing the computation for the complete program when the burdened nodes has completed its work. We consulted with big data experts about the skewness problem but the recommendation to reduce the state being carried during reduce operations was only marginally beneficial. An improved solution was to re-write the algorithm from RDDs to dataframes to take advantage of the SQL Catalyst optimizer. Dataframes are eventually converted to RDDs under the hood but the optimizations by Catalyst are much more efficient than our own refactoring.

Overall, Python, Spark and Hadoop are all well-researched and each have strong support communities. We believe using open source tools broadens the appeal of our algorithm and may help resource-constrained organization implement leading-edge anomaly detection algorithms in their production environments.

5.1.2 Dashboarding and threshold determination

Dashboards are an increasingly popular tool to deliver information to security analysts. The FI uses ElasticSearch, an open source search server, and the kibana plugin for visualization as its dashboard configuration. We intend the alerts generated by our algorithm to appear on the dashboard to alert analysts when highly anomalous shapes occur in the network. The number of alerts raised can easily be adjusted to accommodate the requirements of the security team. Specifically, the team adopts threshold by setting the minimum total shape score to a level that has been historically exceeded the desired number of times in a given day, for example.

When an alert is generated, the analyst will be provided a list of nodes that define the core path or the node generating the star, in addition to the score itself. Further information is available to the analyst if needed. For example, the list of flow counts in the training data that was used to generated the parameter estimates can be compared to the flow count in the current window. If the analyst considers the alert worth investigating, she will know which node to investigate first because the directed nature of our shapes suggest where the anomalous behavior begins. In addition, she know exactly what behavior generated the alert - excess activity on arcs based on historical data. This is an advancement over some machine learning anomaly detection approaches that do not lend themselves to easy interpretation.

5.2 Practical Considerations and Algorithmic Adjustments

We now consider the adjustments the were required to model arcs. First we review Neil's approach to model fitting [34] and then discuss the problems with this approach. Next we describe a set of arcs for which required special handling. Finally, the handling of new arcs is discussed and commented upon.

5.2.1 Neil's approach

Neil, whose research inspired our approach, performed most of his analysis on simulated data with simulated attacks embedded within it despite having access to 'a massive amount of historical network data' that reaches back to late 1990s and contains 'terabytes of data per day' [34]. Our experience working with real-life data suggests his preference for simulated data was driven by a desire to obtain clean and easily interpretable results. There are many non-malicious anomalous events continually occurring in modern network systems and it can be difficult and resource consuming to obtain clear signals. In addition, the variability in arc behavior prevents challenges in the selection of modeling distributions and the handling of new arcs.

When running his application on the simulated data, Neil distinguishes between two arc types using 'the average number of non-zero counts per day, averaged over all 30 days,' and groups arcs into either active or inactive categories based on whether their average is greater than or equal to one, which he considers is an appropriate threshold for fitting distributions. In Neil's data, roughly 45 percent of the arcs met the definition of an active arc. In our test on real data, however, the results were much different. We found less than three percent of data center arcs and less than two percent of office building traffic meeting the active threshold.

Neil then estimates the gamma shape and scale parameters differently for each category of arcs. For active arcs, every arc shares the same scale parameter and, given that scale value, is then fitted with an individual shape parameter. All inactive arcs share the same shape and scale parameters, which are derived by averaging the estimated parameters for the 1,000 most active inactive arcs. We consider these blunt modeling approach to be inappropriate in real network settings because we observe flow counts that differ by six magnitudes among arcs in the FI's network. Given this variability, a common scale (i.e. variance) parameter applied uniformly to all arcs will likely be appropriate to a minority of them, leading to increased anomaly scores when the shared scale is too low and to decreased anomaly scores when the share scale is too high for the arc. The accumulated distortions of the shared scale approach raises considerable questions

about the applicability of Neil's results in production settings. The use of shared scale estimates is not the only weakness of Neil's approach however.

We examined and conducted extensive tests on the distribution of arc flow counts on the FI's network. To our knowledge and that of modeling experts at a national applied science organization, there has been little published research on this topic. Neil selected the gamma because of its useful role in estimating a generalized likelihood ratio test statistic but the gamma scored poorly in fitting tests on our empirical data. Specifically, we fitted 85 distributions, some merely differing in parameter values, to the data and then applied the KolmogorovSmirnov goodness of fit test to rank which ones were the best fit most often. Our results (see section 6.2.1) indicate that Rayleigh distribution provided the best fit for 68 percent of the arcs while the gamma came out as the third best fit for only 21 percent of arcs. Looking at arcs individually, we noticed a wide variety of distribution shapes that made fitting a gamma distribution difficult. We now describe those arcs and how we handled them in the next section.

5.2.2 Problem arcs

As we explained in section 4.5.1.1, the product of the gamma shape and scale parameters equals the mean. In addition, the gamma will always display positive skewness and kurtosis as a function of its shape parameter. We failed to find these characteristics in many of the arcs we observed, and especially for sparse arcs. Two types of arcs were difficult to fit distributions on were ones that exhibited either very low or very high variance.

Low variance arcs, which we defined in our code as those arcs with variance less than three, include arcs that displayed no variance at all. Scipy's gamma fit method will fail when a list of the same values is used as import. Although it can fit a model on arcs with low variance the parameter values make little interpretive sense. For example the shape value will be over 15,000 while the scale estimate will be less than 0.25. These may result in the correct mean value but we hardly have confidence that they model the underlying mechanism.

At the other end of the extreme are arcs that have high variance. In many cases, the fitted variance is caused by flow counts distributed according to a barbell shape. That is, the arc will have a large number of windows with counts in single digit range and another large set of windows with counts 2-3 magnitudes higher. A gamma distribution is clearly not appropriate for these arcs and will generally result in a very low shape parameter to increase skewness and a large negative location value, shifting the entire

distribution to the left so the low and high count windows will have approximately the same (unreasonably low) probability.

A final note about distribution fitting. Scipys fit method maximizes a log-likelihood function that penalizes observations outside of range of the distribution. As mentioned in section 4.5.1.1, an accurate modeling of the tail region is important to calibrate the false positive/negative rate. The fit method's sensitivity to extreme (positive) values in the training set will cause these values, possibly representing malicious attacks, to heavily influence the fitted parameter estimates, right-shifting them along the independent axis. These inflated estimates may result in lower detection rates along those arcs that have been trained on malicious patterns but, again, the localized nature of our algorithm limits the effect this will have on the majority of the network.

5.2.2.1 Adjustments

We handle these problem arcs by first filtering for them and then obtaining descriptive statistics for those windows that have positive flow counts. Then we fit a gamma distribution to the arc activity by setting the shape parameter to 0.5, the location parameter to the 75th percentile of input values and the scale parameter to the mean of the inputs. These parameters will result in the shape depicted in figure 5.2. The 75th percentile was chosen as a concession to barbell shape arc activity distributions while the low shape parameter is a concession to low variance distributions and penalizes observations that are extreme in reference to the 75th percentile value. Finally, by setting the scale parameter to the mean we mitigate the sharp convexity of the distribution shape by accounting the magnitude of values observed on the arc. In the case of underlying barbell shape distributions, this will improve the detection of elevated values in reference to the higher set of flow counts. For low variance arcs, the choice of the 75th percentile will have little influence since low variance represents a tightly grouped set of values. The procedure is the same in the case a normal distribution, except that a shape parameter is not set. Under this construction, a value that is equal to the mean plus the 75th percentile value will be given a 15 percent probability of occurring. A value equal to double the mean plus the 75th percentile value will be given a 5 percent probability of occurring.

We concede that the choice of these parameter values is arbitrary but our experiments with different values failed to noticeably alter the ranking of anomalous arcs within a window. In addition, these values can be easily updated in the code if solid reasons to change them become apparent. We also resisted the urge to complicate the application by handling low variance and barbell distributions using different distributions. Our goal

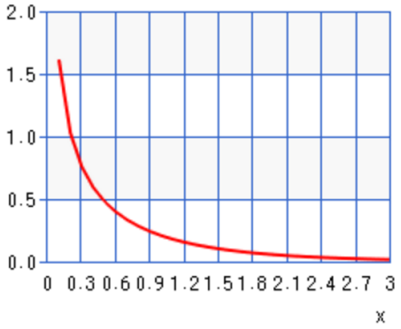


FIGURE 5.2: A Gamma Distribution with Shape = 0.5, Location = 0, Scale = 1

is a lightweight application that is scalable and we are willing to trade some accuracy in a variable environment for speed.

5.2.3 Performance enhancement

We intend our application to be scalable and our design choices reflect this fact. In this subsection we highlight our efforts to balance efficiency with accuracy.

Neil enumerated all anomalous core k-paths and then calculated an anomaly score for the entire shape for only the top one percent most anomalous of these paths [34]. We followed this approach in principle but believed that sorting million of k-paths was unnecessarily expensive. Instead, we first filter out all arcs with Fisher scores below 12, which reflects a probability near 0.25 percent for the observed activity level, before enumerating k-paths and latter calculating shape scores. The removal of more expected arcs should not hamper the effectiveness of our approach because the goal is to find truly rare traversals. What carry out computations on arcs unlikely to impact total shape scores significantly? Security analysts can adjust the threshold value for Fisher scores as necessary to improve application performance as more data is evaluated. Calculating Fisher scores for arcs and storing the values in a dictionary before running the relatively expensive path enumeration algorithm reduced processing time significantly. Stars scores were also calculated using the dictionary. This is to be expected as caterpillar shapes can be conceived as a series of linked stars.

A second deviation from Neil concerns the method of enumerating k-paths. Neil employed a recursive approach shown in algorithm 1. It was written in C and, it is claimed, can enumerate and score 300 million 3-paths in under 5 seconds [34]. In addition, he notes that his algorithm was 'trivially parallelizable' be we challenge this claim in the context of distributed computing tools like Apache Spark.

Algorithm 1 Neil's Recursive k-Path Enumeration Algorithm

```
1: procedure ENUMERATE(E, K)
2:     E ← the list of arcs representing a graph
3:     K ← the integer length of paths to enumerate
4: for each arc A in E:
5:     listp[1] ← A.
6:     RECURSE(E,P,1,K)

7: procedure RECURSE(E, P, L, K)
8:     E ← the list of arcs representing a graph
9:     P ← the list of arcs representing a path
10:    L ← the integer length of P
11:    K ← the integer length of paths to enumerate
12:    arc A = P[L]
13: for each arc B in E:
14:    if A[2] == B[1] then
15:        P[L+1] = B
16:    if L+1== K then
17:        EMIT(P)
18:    else
19:        RECURSE(E,P,L+1,K)
```

As noted in section 5.1.1, Resilient Distributed Datasets (RDDs) are the basic abstraction in Spark. The distributed nature of RDDs requires transformation tasks to be centrally scheduled on multiple slaves. This centralization of scheduling prevents individual executors from launching computations on RDDs, thereby limiting recursive opportunities.

Algorithm 1 also does not account for the sequential order of the path components. Neil does not mention whether the core k-path arcs are checked later to ensure the path order makes sense from a time perspective. If not, then some of the caterpillar shapes may be false positives that waste analysts time.

We experimented with checking sequential order during the k-path enumeration phase but the increase in state caused by carrying the start and end timestamps for every flow along the arc only exacerbated the skewness problem described in section 5.1.1. This was especially true when highly active servers, which can generate tens of thousands of flows in a 30 minute period, were not removed before processing. In some runs of the application, a single node would take over 40 minutes to complete its tasks for a 30 minute window of data, clearly an unacceptable level of performance. Our solution was to enumerate the k-paths and score the shapes without regard to chronology. When the top x results where displayed to the analyst, however, the timestamps were checked. Given that x is generally a tiny fraction of the total enumerated paths, delaying the sequential check saves considerable computational time.

5.2.4 New arcs

As mentioned in section 4.4, new arcs may be a powerful indicator that an attacker is active on the system because his lack of knowledge of normal network behavior results in missteps, including the creation of previously unobserved connections from his controlled computer.

New arcs, however, are a common occurrence on network systems as new computers, or existing computers given new network configuration parameters, join the system. Each node will have a natural rate of generating or receiving new connections that reflects the normal mechanisms underlying its network behavior. We are interesting in detecting the presence of additional mechanisms which may signal that an attack is occurring. In the same way that we consider elevated flow counts as an indicator of suspicious activity along an arc, we consider a rapid increase in the percentage of new arcs created or received as an indicator that anomalous activity is taking place.

We considered two approaches to handling the appearance of new arcs. The first approach is simple: count the number new arcs observed for the node in a window. This count, however, provides the analyst with no baseline for interpreting it and important resource servers will regularly dominate a ranking of nodes experiencing new arc activity. Furthermore, since the data is re-trained only once a day, an arc will remain 'new' for up to 24 hours and again distort the ranking of nodes with high new arc counts. Despite its shortcomings, this was the approach we implemented in our application.

A second approach is discussed in section 7.1.2. That approach uses an asymmetric exponentially weighted moving average approach, which accounts for historical new outarc and inarc generation rates for each arc, uses a simple mechanism to reduce the 'newness' of new arcs and treats previously observed but inactive arcs as 'new' on their reappearance. While more sophisticated, the model incurs space and processing penalties in comparison to the first approach because these new arc generation probabilities must be updated each new window.

5.3 Summary

In this chapter we discussed the implementation details of our algorithms. We highlighted and defended where we differed from Neil, whose work was the inspiration for our approach. In the next chapter, we investigate the assumption of arc independence, the empirical evidence for arc flow count distributions and the sensitivity of shape scores to elevations in flow counts.

Chapter 6

Results

This chapter contains the results of our research. We begin by first discussing the data that was tested and the tools we used to generate results in section 6.1. Our results are provided in section 6.2

6.1 Data Resources and Training

6.1.1 Data

We now describe NetFlow and a brief description of the NetFlow provided to us by the FI.

6.1.1.1 NetFlow

NetFlow is a data collection and aggregation module found on many Cisco routers. The Internet Engineering Task Force (IETF) defines a flow as "a set of packets or frames passing an Observation Point in the network during a certain time interval. All packets belonging to a particular Flow have a set of common properties [55]." These properties can consist of one or more packet header fields, one or more characteristics of the packet itself and one or more of the fields derived from Packet Treatment. Packets belong to the flow for which it completely satisfies all the defined properties of the flow.

Hofstede et al. [2] argue that flow export protocols, including NetFlow, provide advantages over traditional packet capture approaches to network monitoring. These advantages include being well researched, less memory intensive and compliant with existing and future European Union data protection laws. The reduced size of NetFlow records

(on the order of 1/2000 of the original volume) is particularly important to achieve our goal of implementing a scalable algorithm. Despite the reduced memory overhead, however, Big Data tools are often still required to collect and analyze the high volume and speed of modern networks environments.

a. Sample Architecture of a Flow Monitoring Setup

b. Monitoring steps

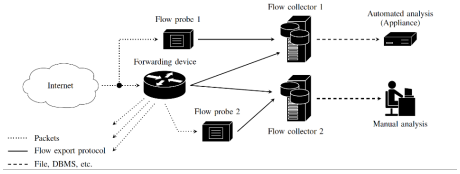


FIGURE 6.1: NetFlow Architecture [2]

Figure 6.1 depicts the common architecture and stages for NetFlow. Packets are captured and preprocessed at an observation point, usually a network interface card in an IP router, in Packet Observation stage. The Flow Metering & Export stage aggregates packets into flows based on common characteristics and then delivers them in a datagram to a flow collector, which receives, stores and preprocesses the data in the Data Collection phase. Finally, in the Data Analysis stage, the data is read into data analysis tools for further analyses.

The Netflow data fields are displayed in figure 6.2. Although limited, the information collected allows for multiple anomaly detection approaches. As described in 3.1 researchers have investigated flow duration, byte size and connection end times as potential indicators of malicious behavior. Academics have pointed out errors in NetFlow's timestamping [56], however, which may introduce noise into time-based detection algorithms. On the whole, though, researchers that have investigated the quality and completeness of NetFlow's collection abilities have been satisfied [57, 58].

Field	Data Type
router_id:	signed int
src_ip:	signed bigint
dst_ip:	signed bigint
packets:	signed int
bytes:	signed int
starttime:	signed bigint
endtime:	signed bigint
srcport:	smallint unsigned
dstport:	smallint unsigned
protocol:	signed tinyint
TCPFlags:	char

FIGURE 6.2: NetFlow Data Types

6.1.1.2 NetFlow data provided by the financial institution

The FI provided NetFlow data collected in 2015 from two primary locations, an office building and a data center. The office building records contains local area network (LAN) traffic that offers the best opportunity to observe network traversals because a large percentage of the network hops are captured in the NetFlow records. Netflow can only collect data that passes through a collection point, in our case a router, although some traffic may be re-directed by a switch before reaching a router and any direct peer-to-peer traffic will not be detected.

The data center traffic was collected at the core router level. This level in a network hierarchy generally handles aggregated traffic from lower levels resulting in considerable LAN traffic to be obscured. The FI data scientists have identified work stations within the records, however, although their exact number is unknown.

6.1.2 Training

Training the data requires estimating distribution parameters for every arc that allows for fitting (see section 5.2 for a discussion of these criteria) and handling those arcs that do not allow for fitting. This is an relatively expensive process even for one medium-sized building that contained more than 600,000 arcs and will become an increasing issue when the application is asked to handle traffic from multiple sources.

We initially trained the data using 10 full days of data and then re-trained the parameters each day at 0100 GMT when we emulated our application over historical office traffic. This allowed 30-minute windows to be processed and scored in 4-5 minutes, which was acceptable given that a new window, accounting for overlaps, was created every 20 minutes. There are several implications of our design choice, which we now discuss.

Re-training the data only once a day implicitly assumes that the mechanisms generating network behavior do not exhibit predictable cycles within the period, an assumption we explicitly reject as network behavior exhibit cyclical patterns throughout the day, the week and the year. For office building traffic, network activity increases when workers arrive in the morning and decreases when they leave in the late afternoon. The same reasoning applies to weekday and weekends. In addition, researchers have noted that differences in network behavior for each day of the week [24] and, of course, national holidays and other special events may contribute to predictable network patterns. Statisticians and econometricians have developed sophisticated models, including the family of autoregressive integrated moving average (ARIMA) models, to account for known cycles and we do not dispute the benefit of implementing them. However, we chose not

to apply them in our research primarily because the LAN traffic available was of limited duration. As mentioned in ref 5.2, we felt an arc needed at least 15 windows of activity before its parameters could be reliably estimated. In 480 30-minute windows of training data, only 34 percent of the arcs met this criteria. Extrapolating this result, we would need 480 days or 480 weeks of data to obtain similar results to develop parameter estimates by hour or by day and hour, respectively. There would be a proportional increase in the amount of state to carry and of computation to carry out. Despite these added factors, we believe accounting for cyclical events should be attempted when the amount of data, storage and processing power is available

A second consequence of our choice to re-train the parameters daily is that new arcs continue to be marked as 'new' throughout the day until the data is retrained. As a result, they distort the list of nodes generating new data and in general will be scored as fairly anomalous when they are in fact quite common when their context is considered. This distortion should decrease over time as the amount of data used to estimate parameters increases. Indeed, if given enough historical data, then the application may re-estimate distribution parameters less frequently than once a day (e.g. Neil trains once a week [34]). As discussed in section 7.1.2, an asymmetric exponentially weighted moving average offers a simple way to adjust a node's natural rate of generating and receiving new arcs over time.

For completeness, we restate the assumption made in section 7.3.1 that there is a strong possibility of attacks existing in the training set. The probability may increase over time as more data is used to train the data but this may be mitigated as more attacks in the historical data are identified. We also note again that the granular view of our algorithm reduces the chance that an attack in the training data will influence the anomaly scores of arcs that are not affected by the attack.

Having finished our discussion of the data and how distribution parameters are trained, we now turn to our results.

6.2 Results

Although no known traversals existed in the traffic available to us, we did test our algorithm on an instance of real malware in the NetFlow data center data. This case involved malware that was detected through proxy logs to be beaconing out to contact a command and control server. As we expected, our method failed to flag this behavior as anomalous because the algorithm is designed to detect network traversals in the action stage of an attack chain (see figure 4.1, while beaconing out occurs in the previous

stage. Furthermore, the beacon message itself is small and does not introduce a large new mechanism on the impacted arc's behavior. In addition, the most anomalous shapes found earlier and later windows around the beaconing events failed to include the node in question.

Without the ability to calculate detection, false positive or false negative rates on real traversal attacks, we explored the independence and distribution of flow counts on arcs and conducted a sensitivity analysis of flow count elevation on shape scores. The next three sections provide the results of our exercises.

6.2.1 Distributions of arc flow counts

As mentioned in section 3.1.3, researchers have investigated various aspects of network behavior (e.g. byte sizes, connection end times, packet counts) but to our knowledge none have looked specifically at the empirical distribution of flow counts along arcs. Neil [34] assumes a gamma distribution for flow counts but provides no empirical support for its use. The choice of distribution is important because modeling the tail section accurately will reduce the amount of false positives and false negatives.

We used the Kolmogorov-Smirnov goodness of fit test on 85 different distributions found in scipy.stats module to see which resulted in the best fit for the empirical data. Several distributions were used more than once but with different shape or degrees of freedom parameter values. The results are displayed in table 6.1. The fits were estimated for July data center traffic and the entire office center traffic.

As can be seen, the Rayleigh distribution was the best fit most often for office building traffic and the best fit the second most often for data center traffic. The Rayleigh is a special case of chi-square distribution with degrees of freedom equal to 2. The chi-square is itself a special case of the gamma distribution with the shape parameter equal to degrees of freedom/2 and scale parameter equal to 2. These results provide support for Neil's and our use of the gamma for modeling flow counts. The normal distribution, also implemented in our application, was the second best fit for most arcs in office building traffic the second best fit the second most often for data center traffic.

The alpha distribution appears frequently in the table. Its pdf formula is shown in equation 6.1, where $a > 0$ is the shape parameter and $\Phi(a)$ is the normal CDF. Note that this distribution as formulated in the python stats module differs from the Lvy alpha-stable distributions of the same name. Its structure has similarities to the normal distribution but we're uncertain to its special properties and use cases.

Month	Fit	Most freq	%	2nd most freq	%	3rd most freq	%
Jan	**Best**	Rayleigh	68.3%	Alpha (9)	20.2%	Alpha (10)	11.5%
	2nd Best	Normal	68.3%	Alpha (10)	20.2%	Alpha (11)	11.5%
	3rd Best	Alpha (9)	46.5%	Gamma (6.75)	21.8%	Alpha (11)	20.2%
July	**Best**	Alpha (7)	48.1%	Rayleigh	22.0%	Double Weibull (3)	13.1%
	2nd Best	Alpha (6.75)	48.1%	Normal	22.0%	Double Weibull (4)	13.2%
	3rd Best	Alpha (6)	48.1%	Chi-Square (1)	21.1%	Double Weibull (5)	13.1%

TABLE 6.1: The best distribution fits for arc flow counts. Each row displays the most frequent distributions that provided the best fit for arc flow counts in the network. The shape parameter (alpha, gamma) or degrees of freedom (chi-square) value is provided in parentheses.

$$alpha.pdf(x, a) = \frac{1}{x^2 \Phi(a) \sqrt{2\pi}} e^{-\frac{(a-1)^2}{2x^2}} \tag{6.1}$$

6.2.2 Arc independence

As mentioned in section 4.5.1.4, the assumption of arc independence has consequences when their true correlation is not zero. For example, if two arcs normally exhibit highly negative correlation, then an increase in flow counts on both arcs should lead to a lower p-value than would be the case if they were truly independent, leading to more false negatives under the independence assumption. The situation is reversed when arcs are positively correlated.

We tested the correlations among arcs over the course of a single week in the office building data. For each day, flow counts for 48 30-minute windows were calculated for each arc, including 0's if no activity occurred on that arc in a window. A Pearson correlation coefficient matrix was then estimated using numpy's corrcoef function. Because the estimates had to be calculated in the driver, small 48 x 48 matrices were used to improve processing time. This amount of arcs only represented a small fraction of the total arcs found during the day so 25 of these matrices were randomly sampled before correlation values were averaged. Even this amount of arcs represents a small amount of the total in the network but we our access to the tools became limited and we believe the random nature of the sampling reduced much of the bias that can be expected of small data sets.

Table 6.2 shows the descriptive statistics of our results by day. Although negatively correlated arcs are present in the daily data, the means indicate that arcs are positively correlated only slightly, with no day exhibited an average correlation greater than 0.084.

	Min	**Max**	**Mean**	**Variance**	**Skewness**	**Kurtosis**
Monday	-0.58	1.00	0.051	0.040	1.775	4.823
Tuesday	-0.36	1.00	0.034	0.041	2.451	7.672
Wednesday	-0.42	1.00	0.057	0.035	2.080	5.034
Thursday	-0.79	1.00	0.034	0.031	2.019	6.206
Friday	-1.00	1.00	0.067	1.067	0.134	-1.982
Ave Weekday	**-0.63**	**1.00**	**0.05**	**0.24**	**1.69**	**4.35**
Saturday	-0.58	1.00	0.084	0.079	2.052	3.502
Sunday	-0.97	1.00	0.037	0.043	1.380	5.762
Ave Weekend	**-0.77**	**1.00**	**0.061**	**0.061**	**1.716**	**4.632**

TABLE 6.2: The correlations among arc flow counts by day

In addition, the correlation distribution exhibits positive skewness and kurtosis in nearly all days, indicating that observed means were righted shifted by a number of highly positively correlated outliers. Thus we should expect that more than half the correlations were actually less than the 0.084 mean. These results suggest that the independence assumption produces slightly elevated levels of false positives but expanded testing is required before definitive statements can be made about the effect of correlations on observed false positive or false negative rates.

6.2.3 Shape score sensitivities

We now examine how sensitive the shape scores found in the FI's network are to elevated flow counts. These elevations theoretically could be generated by an additional mechanism operating in the system, such as a malicious attacker. The goal of this simulation is to observe how much activity is required before the shape's score causes it to be the most anomalous in the window, attracting the attention of the security analyst for further investigation. In addition, we explore whether the algorithm is more sensitive to increased activity during periods of low network traffic. If so, then modifications to the algorithm (e.g. lowering the shape parameter values) may be required for periods of higher activity.

We trained the algorithm on office building data for sixteen days and then examined two 30 minute windows, one at 07:00 and the second at 14:00, on the next day, a Thursday. These windows represent low activity and high activity periods, respectively. Earlier windows were considered for the low activity period but rejected because they generated very low amounts of caterpillar shapes with a core path of length three. We believe the findings of the 07:00 window will be even more pronounced in earlier, lower activity windows in the day. Note that known scanners, application discovery servers

and other generators of non-malicious anomalous activity were filtered out before the analysis. This made the data cleaner and the filter could easily be applied in a production environment.

For each window, the shapes were ranked in descending order by their total Fisher score and the top 100,000 were selected. Within this subgroup the shapes that appeared at the 20%, 40%, 60%, 80% and 100% (i.e. the last) levels were chosen to examine how increases in flow counts affected subsequent rankings. We increased the absolute level of arc flow counts by seven amounts (0, 2, 7, 20, 54, 148, 403), which correspond to $int(e^x) for x \in 0-6$. We felt this method would provide an appropriate range to data point ratio. The 0 data point provided a sanity check on our results.

In addition to increasing the flow count on the core path arcs in each instance, we also raised a percentage of the outarcs emanating from the core path nodes. These outarcs can be considered scanning behavior by the attacker after she has infected a node and looking for the next hop in the traversal. The percentages start at 0% (just a traversal along the core path with no additional scanning) up to 100% (increasing activity on every outarc possible) by increments of 20%.

We made some concessions to handle new arcs in our sensitivity testing. Because new arcs do not have gamma parameters in the training set, they receive a probability equal to the percentage of new arcs to total arcs in that window, which is then used to calculate that arc's Fisher score. Thus increasing the flow counts on these new arcs will have no effect on the resulting Fisher score. Therefore, we (1) eliminated all shapes containing core paths with new arcs before ranking them and (2) excluded new outarcs from the set of outarcs whose flow counts were elevated. For example, when 20% of the outarcs had their counts increased, this percentage applied only to previously known arcs in the shape and may have resulted in less than 20% of the total outarcs being selected. We did not investigate the true percentage of outarcs that were elevated but we note that the percentage of new arcs was observed to be less than one percent after ten days of training.

Table 6.3 shows the Fisher scores for the shapes found at the selected levels and the separation between the levels. We note that the amount of separation between three of the levels is quite small (less than 150) in both activity windows, which suggests that rapid increases anomalous ranking could be made without large increases in arc activity. There does not seem to be material differences in the shape scores between the two time periods. This implies that our algorithm should detect elevated arc flows equally well throughout the day.

Shape Rank	Low Activity Window		High Activity Window	
	Fisher Score	**Difference**	**Fisher Score**	**Difference**
First	5405.13	-	5405.13	-
20%	4716.10	689.03	4529.01	876.12
40%	4588.57	127.52	4400.33	128.68
60%	4449.25	139.33	3678.75	721.58
80%	3905.65	543.59	3490.21	188.54
100%	3763.13	142.52	3426.45	63.77

TABLE 6.3: Shape scores and their separation during the low activity window.

Tables 6.4 and 6.5 confirm this conjecture, in the broad strokes at least. In both periods, elevating the core path itself without scanning activity on the edges failed to raised the shape score to a level that would be noticed by analysts. Once scanning behavior begins, however, the increased activity soon rises to the most anomalous shape score in the window, indicated by their 0.001% ranking (recall that a subset of 100,000 shapes was used). Even the least anomalous shape in our set rises to the most anomalous rank when 40% of the known arcs are elevated by 20 flows per arc in both periods. The 40,000th most anomalous arc rises to the most anomalous increasing known outarcs by just 7 flows in 20% of 'fuzz hairs' in the low-activity period and in 40% of the hairs in the high-activity period. These results suggest that moderate to heavy scanning behavior by an attacker traversing the network could be flagged as anomalous by our algorithm.

6.3 Summary

The chapter discussed the data we used and the findings we obtained from them.

Orig rank	Fuzz arcs elevated	Elevated Flow Count						
		0	**2**	**7**	**20**	**54**	**148**	**403**
20	**0%**	20.000	15.050	9.719	2.561	0.238	0.086	0.054
	20%	20.000	0.240	0.001	0.001	0.001	0.001	0.001
	40%	20.000	0.001	0.001	0.001	0.001	0.001	0.001
	60%	20.000	0.001	0.001	0.001	0.001	0.001	0.001
	80%	20.000	0.001	0.001	0.001	0.001	0.001	0.001
40	**0%**	40.000	34.031	26.011	10.046	0.797	0.240	0.240
	20%	40.000	1.000	0.001	0.001	0.001	0.001	0.001
	40%	40.000	0.004	0.001	0.001	0.001	0.001	0.001
	60%	40.000	0.001	0.001	0.001	0.001	0.001	0.001
	80%	40.000	0.001	0.001	0.001	0.001	0.001	0.001
60	**0%**	60.000	57.423	53.568	43.612	18.579	4.801	1.030
	20%	60.000	6.239	0.001	0.001	0.001	0.001	0.001
	40%	60.000	0.080	0.001	0.001	0.001	0.001	0.001
	60%	60.000	0.001	0.001	0.001	0.001	0.001	0.001
	80%	60.000	0.001	0.001	0.001	0.001	0.001	0.001
80	**0%**	80.000	78.624	76.617	73.605	71.315	70.778	70.601
	20%	80.000	71.189	8.784	0.001	0.001	0.001	0.001
	40%	80.000	37.303	0.001	0.001	0.001	0.001	0.001
	60%	80.000	2.154	0.001	0.001	0.001	0.001	0.001
	80%	80.000	0.025	0.001	0.001	0.001	0.001	0.001
100	**0%**	100.000	95.122	92.917	88.497	80.669	73.910	71.758
	20%	100.000	77.580	69.858	0.149	0.001	0.001	0.001
	40%	100.000	72.811	15.263	0.001	0.001	0.001	0.001
	60%	100.000	71.157	0.055	0.001	0.001	0.001	0.001
	80%	100.000	66.639	0.001	0.001	0.001	0.001	0.001

TABLE 6.4: Low-activity period sensitivity of shape score ranking to increased flow activity along core k-path arcs and adjacent outarcs. The numbers in the table represent the new ranking within the window after the edge flow count has been increased by the values list in the rightmost columns. Core arcs are elevated in every instance. A percentage of previously observed outarcs, indicated in the 'Fuzz arcs elevated' column, are also elevated by the same values.

Orig rank	Fuzz arcs elevated	Elevated Flow Count						
		0	**2**	**7**	**20**	**54**	**148**	**403**
20	**0%**	20.000	16.101	15.182	13.281	9.169	3.258	1.432
	20%	20.000	3.227	0.031	0.001	0.001	0.001	0.001
	40%	20.000	1.158	0.001	0.001	0.001	0.001	0.001
	60%	20.000	0.271	0.001	0.001	0.001	0.001	0.001
	80%	20.000	0.026	0.001	0.001	0.001	0.001	0.001
40	**0%**	40.000	37.072	35.803	32.578	22.259	5.503	1.361
	20%	40.000	11.908	0.226	0.001	0.001	0.001	0.001
	40%	40.000	2.827	0.001	0.001	0.001	0.001	0.001
	60%	40.000	1.234	0.001	0.001	0.001	0.001	0.001
	80%	40.000	0.122	0.001	0.001	0.001	0.001	0.001
60	**0%**	60.000	58.055	57.370	55.351	48.915	44.662	44.503
	20%	60.000	46.704	44.148	0.002	0.001	0.001	0.001
	40%	60.000	44.567	1.020	0.001	0.001	0.001	0.001
	60%	60.000	44.356	0.001	0.001	0.001	0.001	0.001
	80%	60.000	36.703	0.001	0.001	0.001	0.001	0.001
80	**0%**	80.000	73.630	69.291	63.134	47.609	44.658	44.522
	20%	80.000	57.736	44.361	0.005	0.001	0.001	0.001
	40%	80.000	46.559	2.673	0.001	0.001	0.001	0.001
	60%	80.000	44.577	0.005	0.001	0.001	0.001	0.001
	80%	80.000	44.361	0.001	0.001	0.001	0.001	0.001
100	**0%**	100.000	93.639	87.754	76.856	64.245	45.099	44.667
	20%	100.000	68.215	45.567	3.144	0.001	0.001	0.001
	40%	100.000	57.613	41.480	0.001	0.001	0.001	0.001
	60%	100.000	47.903	1.594	0.001	0.001	0.001	0.001
	80%	100.000	45.124	0.009	0.001	0.001	0.001	0.001

TABLE 6.5: High-activity period sensitivity of shape score ranking to increased flow activity along core k-path arcs and adjacent outarcs. The numbers in the table represent the new ranking within the window after the edge flow count has been increased by the values list in the rightmost columns. Core arcs are elevated in every instance. A percentage of previously observed outarcs, indicated in the 'Fuzz arcs elevated' column, are also elevated by the same values.

Chapter 7

Discussion and Future Work

In this chapter we discuss important issues regarding our research and suggest productive areas for future work. In the interest of clarity, we have grouped these topics into theoretical, practical, financial institution, data and results categories.

7.1 Theoretical Issues

This section will look at known issues of the algorithm from an abstract design perspective.

7.1.1 Arc independence

Our detection algorithm assumes the independence of arcs and while we found some positive correlation among arcs in our research (see section 6.2.2), we encourage practitioners to test for arc independence in their systems before implementing our approach as every network will exhibit different behavior. If a stronger correlation among arcs was detected, there exist other methods of combining p-values, such as Brown's method [59], that could account for these dependencies. Maintaining and updating the variance-covariance matrix of the network may become burdensome on the system, however, and such costs may reduce the appeal of our algorithm, which is designed to be lightweight and scalable.

7.1.2 New arcs

In dynamic systems, new computers and other nodes are continuously added and removed in the network. This trend can be expected to increase as applications migrate

to mobile devices and corporations expand bring-your-own-device (BYOB) programs. These new nodes and resulting new arcs may reduce the power of new arcs to indicate malicious behavior. As noted in section 4.4, one disadvantage for an attacker is that they may not know the normal behavior of the network and naive scanning behavior may abnormally increase the amount of new arcs emanating for a node. Thus accurately detecting anomalous new arcs can be a powerful approach to finding attack patterns in the network.

One way to dynamically model the appearance of new arcs through time is using an asymmetric exponentially weighted moving average (AEWMA) [4]. Some time after its first appearance a new arc ceases to be 'new'. Similarly, an arc that fails to be active for a specified length of time should not be considered active and its reappearance can be regarded a 'new'. AEWMA captures this behavior in a way that is lightweight and efficient.

A Bernoulli model 4.7 provides the estimate for the probability of a new arc. The AEWMA estimate is updated as follows:

$$p_{new} = [\lambda_1 + (1 - \lambda_1)p]X_t + (1 - \lambda_2)p(1 - X_t) \tag{7.1}$$

where p is the historical probability of a new $X_t = 1$ if the arc is observed in the current window while $X_t = 0$ if it is not. λ_1 represents the rate the parameter increases over time while λ_2 represents the rate the parameter decreases over time. The values of λ_1 can be set so the appearance of highly anomalous arcs will remain so for a limited period and the value of λ_2 can be set so a non-anomalous arc will be categorized as anomalous in its next appearance after a large amount of time.

The AEWMA approach does require that more state be carried because the historical new outarc and inarc probabilities must be referenced but the space and time costs are linear in the worst case. In addition, the theory behind new arcs as a detection algorithm has not be demonstrated in production environments in the literature and we hope that more research is conducted in this regard.

7.1.3 Flow count distribution modeling

The literature we reviewed did not contain empirical evidence of the most appropriate distributions to model arc flow counts. Selecting the correct distribution, and especially how right tail section confirms to observed data, is critical to reducing false negatives and, more importantly, false positives, which can distract an analyst's time from investigating true malicious anomalies.

We implemented both the gamma and normal distributions in our application and the results in section 6.2.1 provide some support that these are appropriate choices. Several edges in the FI data, however, were difficult to fit a distribution to because they appeared only once, they consistently had the exact flow count in every window of activity or their histogram exhibited a barbell shape. For these arcs, we set the location parameter to the 75th percentile of input values and the scale parameter to the mean of the inputs for both our gamma and normal distribution implementations. For the gamma distribution, we also set the shape parameter to 0.5, which heavily penalizes outliers. While these choices seem logical to us, we concede that other values could also be considered. The lack of real traversal attacks prevented us from varying these parameters to calibrate the system to real-world malicious anomalies. That said, we believe our approach is superior to that of Neil [34], who first separated highly active arcs from less active arcs and then applied one scale value to each set and one gamma shape parameter value to all less active arcs. This approach doesn't consider the wide variability of arc flow counts, which can range from single digits on some arcs to six digits on other arcs. In addition, no accommodation is made for arc flow counts exhibiting a barbell shape. We believe his design choices were heavily influenced by the use of simulated network traffic instead of real-world traffic.

We estimated our distributions by day, making no distinction between weekdays/weekends, work/non-work hours, holidays/weekends and seasonality we observed these patterns in the FI NetFlow data. In addition, normal arc activity may evolve as employees gain new responsibilities or respond to new pressures. Statisticians have created models to account for these cyclical patterns and shocks to the system. A common approach is to use an autoregressive integrated moving average (ARMA) model, which captures regular patterns in the data but may not account for special events, such as holidays. Like implementing the AEWMA model for new edges, the use of ARIMA models would increase processing costs and require a small amount of state to be stored.

7.1.4 Sessions

Our approach only considers uni-directional traffic but most communication in a network is bi-directional. Counting the flows in a session, which are grouped arcs of the same conversation, is an alternate approach to our algorithm that may reduce false positives by aggregating arcs into logical sets. The creation of sessions involves some guesswork as to which arcs belong to which sessions but the use of sessions provides a better abstraction for the underlying behavior we're modeling. The algorithm would still seek to detect star and caterpillar shapes but one might gain increased confidence that these

shapes are generate by the same mechanism or actor. We encourage more work in this regard.

7.1.5 Subnets

An alternate use of our algorithm was suggested by a security architect at the FI. She suggested that because the network is segmented along logical lines the arcs of most interest could be those that traverse subnets. This an approach that is similar to Aggarwal [32] but applied to the network intrusion domain. We believe this idea holds some promise but our algorithm was designed to capture traversals across several nodes. Perhaps a simpler scan statistics model could be used in conjunction with our approach for a more augmented view of an attack.

7.1.6 Design choices: window size, path size and shapes

The choice of window size and the length of the core path both have consequences in the effectiveness of the algorithm. Some attackers may be impatient, hoping to complete a traversal and exfiltrate data within a few minutes, while others may be very patient, taking days or weeks to accomplish their goal while attempting to evade detection. The use of different window sizes in a production environment is encouraged to capture a range of attack methods. More research is needed to find the window sizes that provide the highest probability of capturing modern traversal attacks.

A second design choice is number of arcs to include in the body of the caterpillar shape. The correct number likely depends on the production environment and historical attacks found in data could provide a useful guide. We followed Neil [34] in choosing a path length of three. Two hop attacks should also be captured using our method but the anomaly score will likely be less pronounced. The case is similar for traversals of four or more hops. It's likely that considerable attack data is required before the most efficient choice can be made with confidence.

Finally, the star and caterpillar shapes we aimed to detect does not exhaust the possible network shapes indicating a traversal attack. Stars, in particular, would seem to capture an unsophisticated actor scanning loudly around the machine he compromised. Advanced persistent threat actors are often very patient and are careful to not generate noticeable anomalies in the network. Continued research may reveal more promising and indicative attack shapes.

7.1.7 Feature enrichment

Our implementation only considered flow counts along arcs using Transmission Control Protocol (TCP) for our scan statistic approach but adding more NetFlow characteristics (see figure 6.2) to the analysis may improve anomaly detection. The most effective features or other communication protocols to consider adding to (or replacing) TCP flow counts is an open question that requires more research.

7.1.8 The duplication problem

A single node may be a member of numerous subgraphs, including some with considerable overlap with others. This allows multiple concurrent attacks to be detected but also may cause one attack to generate multiple high-ranking subgraphs, obscuring more subtle attacks in the same period. We're uncertain the best way to de-duplicate subgraphs containing the same attack while still maintaining the ability to detect multiple attacks from the same node. Grouping flows into sessions, described in section 7.1.4 offers one possible.

7.2 Practical Issues

We now look at four practical issues that relate less to algorithm design and more to limitations on current platforms and tools.

7.2.1 Path enumeration

The appropriate enumeration of directed core three paths raised several issues. NetFlow groups packets into flows using a classification method that is not always correct, which introduces some error into our algorithm. A more troubling problem is determining how long a time overlap must exist between flows before they can be considered for a core path construction.

Figure 7.1 illustrates the problem. The nodes in the path are numbered in the order that might appear in the directed core path. Notice that a brief overlap in time exists for attacker traffic to be captured in these flows to complete a three path. It's an open question how long an attacker needs to 'jump' flows successfully. In addition, an arc may have thousands of flows in a 30-minute window that meet the criteria path enumeration. This increases the computational cost of the algorithm and a better definition of

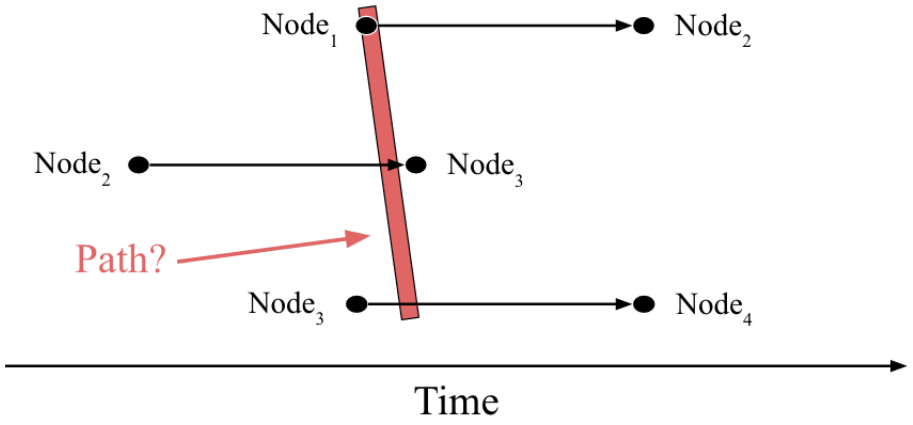


FIGURE 7.1: A questionable path

what constitutes a likely path would improve the efficiency and the effectiveness of the approach.

7.2.2 Port 0

We noticed many flows with the unusual port number of zero. A consultant working at the FI that NetFlow will separate TCP communications longer than 5 minutes into separate flows, which can be identified because the source port is '0'. In addition, packets that exceed the maximum transmission unit (MTU) size are fragmented into several packets but only the first packet will contain an valid TCP port. The remaining fragments will have no layer 4 header and thus have a destination port set to 0 [60]. We filtered out flows with this port value because the first packet should already been captured and recombining these flow with their logical pairs would be a laborious process. Packets containing a port number of zero may indicate more malicious activity, however.

IANA's Service Name and Transport Protocol Port Number Registry list port 0 as reserved, but valid, for both TCP and UDP[61]. Because the specification does not define behavior for connections established on those ports, attackers may use responses to fingerprint the operating systems of destination hosts. Furthermore, hackers may craft 'impossible' packets to DDoS firewalls because some routers prevent administrators from entering port 0 in the access control list since it's supposedly impossible for traffic to be on that port. Making such packets requires using raw sockets software calls that specify everything after the Ethernet header using bytes[60]. We don't believe, but cannot be sure that either of these activities were occurring in the FI network.

7.2.3 NetFlow taps

NetFlow can only record the traffic that is routed through the collection point, limiting the visibility of the entire network by the tool. Any traffic that is peer-to-peer or handled by a switch will not be detected. This is a limitation of our approach's focus on network flows. Augmenting the NetFlow data with other collection sources, including proxy logs, DNS requests and Unified Host Collection Agents, may close some of the gaps in the NetFlow records.

7.2.4 Distribution fitting

As mentioned in section 5.2.2, the fit method found in the scipy gamma and normal modules is sensitive to extreme (positive) values in the training set and will shift the distribution to the right when anomalously high flow counts are observed. These values may represent a malicious attacker operating on the system but even when those counts have non-malicious origins, the resulting right shifted distribution increase the false negative rate.

Examining the correctness of the tail region probabilities is important considering our anomaly scores are based on the rarity of occurrences in this region. The Anderson-Darling test, which is a modification of the Kolmogorov-Smirnov goodness-of-fit test, is useful method to check the these probabilities. It does require that distribution parameters be specified beforehand so checking millions of edge distributions may involve considerable resources.

7.3 FI Issues

In this section, we briefly discuss the issues related to the FI's approach to data science. Section 7.3.1 noted the lack of a data catalog that would allow the training set to be free attacks and non-malicious anomalies. The FI has begun this cataloging process and we believe this will improve confidence in algorithmic results. A catalog of real attacks in the NetFlow records would help gauge the effectiveness of different applications and tools in the FI environment.

Until a set of real attacks in NetFlow records is collected, the FI might consider the configuration of a honeypot subnet whose IP addresses have not been broadcast to the network. This approach as a tool to improve anomaly detection was suggested by Stalmans and Irwin [62] and is based on the notion that any connection attempts to this subnet is likely malicious.

The FI changed their NetFlow collection taps from the local access level in the office building to the core router level at a data center to better capture malware beaconing out to command and control centers for instructions, among other reasons. The decision was a conscious one to reduce the risk of exfiltration across the network perimeter but it does increase their exposure to internal attacks that have no need to cross proxy servers. The prioritizing of risks is the nature of business risk management but the increase in reported insider attacks (see section 1.1.1.1) argues for continued vigilance of the internal environment.

7.3.1 The data

As Lee and Stolfo have noted [63], when collected data is not designed specifically for security purposes or can not be used directly to build a detection model, a considerable amount of (iterative) data pre-processing is required. We experienced some of the same issues with the provided Netflow data. First, no data catalog existed for the data under examination. This means nodes were not labeled, scheduled network events were not documented and known attacks were not listed. To understand the problems caused by this lack of information, consider non-anomalous computers, including vulnerability scanners and IBM's Tivoli Application Dependency Discovery Manager (TADDM) servers [64], that often generate highly anomalous traffic that must be removed before more subtle anomalies present themselves. We filter these nodes before running our algorithm to reduce noise but note that the decision to exclude source addresses and source ports limits the attack space on the basis of prior information. One could argue that such filtering amounts to a semi-supervision approach to anomaly detection, and we would agree. It is clear, however, that unfiltered flows generates too much noise for our localized ranking approach to be useful in a production environment.

Cleaning the trial data of non-malicious anomalous network events is relatively easy but removing existing attacks in it was difficult to impossible. The FI security analysts suspected malware and penetration tests existed in the data but the exact nodes affected and the time range were unknown. At least one port scan was discovered by the data scientists in the data center traffic but it did not appear anomalous in our algorithm, which does not consider source ports. The FI is aware of benefits of labeling the data and have made promising strides toward creating and maintaining a data catalog to improve research results.

An additional problem concerned office building data. Specifically, the NetFlow configuration did not collect TCP flag data within the flow records. While our approach is intended to require minimal information, we believe checking the TCP flags set during

a flow could provide useful forensic information and could possibly be added as an enhancement. In addition, the amount of office traffic collected proved to be only 19 days over the course of January and February, which is limited time to train and observe an attack.

These complications have implications for the reliability of trained parameter estimation. We don't know if malicious behavior exists within the ten days training data, which may make 'attack' behavior appear normal. One advantage of our focus on highly-localized subgraphs is that attack behavior in one part of the network is likely to affect only a small number of subgraphs. Unless an abnormally high number of nodes are infected, we believe our approach still has significance for anomaly detection even with the presence of malicious behavior in the training set.

One final potential issue is that the provided data only represents a small fraction, 2-3 percent, according to the FI data scientist. If this data is not representative of behavior patterns existing on other parts of the network, then the reliability of our findings may be limited.

7.4 Our Results

In this section we elaborate on the results we presented in chapter 6. We address each in the order of the research questions they answer.

7.4.1 Distributions of arc flow counts

The distributions that were fitted to arc flow counts nor our choices for parameter values did not exhaust the possible choices. In addition, we believe repeated sampling over the course of the year is necessary to confirm the stability and generality of our results.

Time considerations prevented us from performing tests on the tail region of the data to ensure this critical section is modeled correctly. As mention in section 7.2.4, applying the Anderson-Darling test to our fitted distributions would increase our confidence in their relative ranking for the purposes of our approach.

7.4.2 Arc independence

The method used to obtain the correlations in table 6.2 made a general statement about arcs found in the network as a whole but did not consider the 'nearness' of the arcs. The correlations of arcs that are adjacent to each other in an identified shape, however,

is of most importance to our traversal attack model. We regret that time considerations prevented us from examining correlations at the same localized level as our algorithm so that we could make stronger statements arc independence. Even if time wasn't a consideration, however, the level of confidence we could obtain would be constrained by our level of confidence that the examined data was free of attacks. We also could have calculated the average eccentricity of each node in the examined windows to see how far, on average, each node is from the node most distant from it in the graph. A small mean eccentricity may slightly raise our confidence in arc correlations in the general case could be applied arc correlations in the adjacent case, with full confidence obtained when mean eccentricity equals one.

We did not notice a large difference between weekday and weekend arc correlations. Time prevented us from testing work hours against non-work hours but we suspect that any large high activity period vs low activity period differences would be evident in weekday/weekend numbers. Then again, we allow for the possibility that higher (or weaker) correlations occurring in the typical 8 hour work period, which only represents 1/3 of the total daily weekday traffic, may be muted by the 16 hour non-work period correlations. We hope future researchers shed further light on these differences.

We also would have liked time to explore some of the stronger negative correlations to learn if we could determine a mechanism that could account for them. The Friday data contained at least two arcs that exhibited a perfect negative correlation while at least two arcs in the Sunday data came close at -.97.

Finally, we remind readers that the listed correlations where obtained using only one day of data for each day of the week. Repeated sampling of data at regular intervals may yield different results.

7.4.3 Shape score sensitivities

We tested the sensitivity of elevated flow counts on a set of the 100,000 most anomalous shapes found in the two windows under examination, which are than 10% of the total shapes in each case. We did this because we initially believed lower ranking shapes would have difficulty rising to the top of the list but our results show that expanding our scope to include all shapes may be warranted. We note, however, that raising 20% of the outarc flow counts by 54 raised the total shape score by over 5000 in most cases so even a shape with a initial total score of 1 would still almost reach the top 0.1%, at least in the low activity period. A moderately higher flow count elevation or higher percentage scan should place rank it first in both periods.

Without the ability to observe real traversal attacks in the NetFlow records, however, we cannot confidently state how attacker behavior elevates flow counts. Two additional flows per arc in a 30 minute period seems intuitively reasonable but how plausible is the creation of 20 extra flows, or 54? We hope researchers with access to real traversal attacks can provide information in this regard. It may be the case that different attack objectives result in different flow count elevations, suggesting the amount of elevation could be used as an attack classifier.

The handling of new arcs affected the design and results of our sensitivity test. In our implementation, new arc are considered 'new' for the entire first day they are observed and the reappearance of previously seen but inactive arcs are not considered 'new', but probably should be. We've mentioned in sections 7.1.2 and 4.4 that new arcs have the potential to be powerful indicators of attack behavior. To accommodate our handling of new edges, we filtered out shapes that contained new arcs in their core path and excluded new outarcs from our scanning percentage sets. We would interested how much the sensitivities would change if new arcs probabilities were estimated using the AEWMA method discussed in section 7.1.2. If the differences were significant, then practitioners may be inclined to take on the additional processing and memory costs required by that method.

As was the case for all our findings, we stress that we obtained our results by looking at only a few snapshots of the network. Repeated sampling and testing over a wide range of time and conditions is necessary to increase our confidence in the reliability of the results.

7.5 Summary

This chapter provided a critical discussion on the project, our implementation, the production environment of the FI, the data work with and our findings. Our intention was to provide the reader context for interpreting our results and to highlight some of the limitations of our algorithmic approach. In the next chapter we review our research questions and discuss how our results provides answers to them.

Chapter 8

Conclusions

The main research question this thesis considered was how effective a flow-based anomaly detection algorithm using summary statistics was in a large production environment. To answer this main question, four sub-questions were answered:

RQ 1 - What is the state of the art according to the research literature on network anomaly detection?

We examine the latest research into anomaly detection techniques in section 3.1. The literature highlights numerous approaches to anomaly detection but eventually focused on the scan statistic approach of Neil [4], which was attractive due to its lightweight implementation, its efficiency and its localized perspective. We believe this approach shows the way forward as the massive growth of available data meets improved Big Data tools to process them and anticipates imminent privacy regulations. For these reasons, we chose it as the basis of our own detection model to implement.

RQ 2 - What is the empirical evidence for the distribution of arc activity as measured by flow counts?

Neil appeared to select the gamma distribution to model windowed arc flow counts out of convenience and his results were applied to simulated data. Real production networks are much more complex and we sought empirical evidence on flow count distributions for our own model.

Our results, presented in section 6.2.1, reveal that special cases of the gamma distribution (Raleigh and chi-square) as well as the normal distribution were among the best fits in both office building and data center traffic. This justifies our us of both models in our implementation. The use of the gamma over its special case distributions is due to the former's increased entropy, which minimizes the amount of prior information built

into the distribution. In addition, many physical systems tend to move towards maximal entropy configurations over time. We note that the normal distribution is also maximum entropy among all real-valued distributions with a specified variance.

RQ 3 - What is the empirical evidence for the independence of windowed arc flow counts?

Our algorithm assumes the independence of adjacent arc flow counts, which greatly reduces the amount of computation required and the amount of state that needs to be stored. If this assumption is incorrect and correlations do exist, however, then the algorithm will produce an excess of false positives or false negatives depending on the direction of the correlation.

Our results for correlations is presented in section 6.2.2. The arcs in general do exhibit a positive correlation on average but the magnitude is relatively low (0.05 for weekdays) and the positive skewness suggests that the majority of correlations are even less than this value. More work is needed to determine the correlation among adjacent arc flow counts, however. If those correlations are in line with the general correlations, then it supports our assumption of arc activity independence and suggests that the costs of added overhead required to properly account for these correlations would exceed the resulting benefits in accuracy.

RQ 4 - How sensitive is the algorithm to increases in flow counts along the core path and adjacent outarcs from the core path?

A critical measure of the effectiveness of our approach is the sensitivity of shape scores to flow count elevations along the core path and outarcs emanating from core path nodes. We tested seven different levels of elevation and five different percentages of affected outarcs and noted the resulting rank of that shape's score among the top 100,000 most anomalous shapes found in that time window.

We presented our findings in section 6.2.3. The results show our algorithm quickly detects traversals with scanning behavior but performed less well when scanning didn't take place. Even the 100,000th ranked shape in the window rose to the most anomalous provided at least 20 flows were added to normal values and 40% of the outarcs were scanned.

Main research question: How effective is a flow-based anomaly detection algorithm using summary statistics in a large production environment?

We were unable to accomplish our initial goal of obtaining detection and false negative/positive rates for real traversal attacks in production environments but we verified

important assumptions and design choices empirically and demonstrated the effectiveness of our algorithm in raising the ranking of shapes that have additional mechanisms elevating flow counts on their constituent arcs. These are promising results that suggest that continued research on the algorithm's effectiveness as anomaly detection application is warranted. We hope that research is conducted soon to meet the increasing reports of internal attacks.

Bibliography

[1] Leman Akoglu, Hanghang Tong, and Danai Koutra. Graph based anomaly detection and description: a survey. *Data Mining and Knowledge Discovery*, 29(3):626–688, 2014.

[2] Rick Hofstede, Pavel Celeda, Brian Trammell, Idilio Drago, Ramin Sadre, Anna Sperotto, and Aiko Pras. Flow monitoring explained: from packet capture to data analysis with netflow and ipfix. *Communications Surveys & Tutorials, IEEE*, 16 (4):2037–2064, 2014.

[3] Richard Lippmann, Kyle Ingols, Chris Scott, Keith Piwowarski, Kendra Kratkiewicz, Mike Artz, and Robert Cunningham. Validating and restoring defense in depth using attack graphs. In *Military Communications Conference, 2006. MILCOM 2006. IEEE*, pages 1–10. IEEE, 2006.

[4] Joshua Charles Neil. Using new edges for anomaly detection in computer networks, June 25 2015. US Patent 20,150,180,889.

[5] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15, 2009.

[6] Charu C Aggarwal. *Outlier analysis*. Springer Science & Business Media, 2013.

[7] Douglas M Hawkins. *Identification of outliers*, volume 11. Springer, 1980.

[8] Animesh Patcha and Jung-Min Park. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer networks*, 51(12):3448–3470, 2007.

[9] Karen Scarfone and Peter Mell. Guide to intrusion detection and prevention systems (idps). *NIST special publication*, 800(2007):94, 2007.

[10] Lawrence E Bassham and W Timothy Polk. Threat assessment of malicious code and human threats (nistir 4939). *National Institute of Standards and Technology Computer Security Division*, 1992.

[11] Frank L Greitzer, Andrew P Moore, Dawn M Cappelli, Dee H Andrews, Lynn Carroll, Thomas D Hull, et al. Combating the insider cyber threat. *Security & Privacy, IEEE*, 6(1):61–64, 2008.

[12] Internet crime complaint center (ic3) — increase in insider threat cases highlight significant risks to business networks and proprietary information. http://www.ic3.gov/media/2014/140923.aspx, Sept 2014. (Visited on 11/19/2015).

[13] www.pwc.com/us/en/increasing-it-effectiveness/publications/assets/2014-us-state-of-cybercrime.pdf. http://www.pwc.com/us/en/increasing-it-effectiveness/publications/assets/2014-us-state-of-cybercrime.pdf, Jun 2014. (Visited on 11/19/2015).

[14] Ross Anderson. *Security engineering*. John Wiley & Sons, 2008.

[15] Danny Yadron. Symantec develops new attack on cyber-hacking - wsj. http://www.wsj.com/news/article_email/SB10001424052702303417104579542140235850578-lMyQjAxMTA0MDAwNTEwNDUyWj, May 2014. (Visited on 12/01/2015).

[16] Ian Barker. Zero-day vulnerabilities increase over 2014. http://betanews.com/2015/08/06/zero-day-vulnerabilities-increase-over-2014/, Aug 2015. (Visited on 12/01/2015).

[17] Cisco. Introduction to cisco ios netflow - a technical overview - cisco. http://www.cisco.com/c/en/us/products/collateral/ios-nx-os-software/ios-netflow/prod_white_paper0900aecd80406232.html, May 2012. (Visited on 10/26/2015).

[18] Ben Rossi. New eu data laws go-live date finally revealed and why its costs will run into the billions — information age. http://www.information-age.com/technology/information-management/123459991/new-eu-data-laws-go-live-date-finally-revealed-and-why-its-costs-will-run-billio Aug 2015. (Visited on 12/01/2015).

[19] Article 29 Working Party. Protection of personal data - european commission. http://ec.europa.eu/justice/data-protection/index_en.htm. (Visited on 12/01/2015).

[20] Cal-Irvine Information and Computer Science. Kdd cup 1999 data. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, Oct 1999. (Visited on 09/16/2015).

[21] David Shamah. Israeli start-up claims it may be able to stop all viruses — the times of israel. http://www.timesofisrael.com/hack-this-start-up-claims-it-can-stop-all-viruses-permanently/, Feb 2014. (Visited on 11/18/2015).

[22] Dambisa Moyo. Will banks survive digital disruption? - agenda - the world economic forum. https://agenda.weforum.org/2015/11/will-banks-survive-digital-disruption/, Nov 2015. (Visited on 11/20/2015).

[23] Dorothy E Denning. An intrusion-detection model. *Software Engineering, IEEE Transactions on*, (2):222–232, 1987.

[24] Monowar H Bhuyan, Dhruba Kumar Bhattacharyya, and Jugal Kumar Kalita. Network anomaly detection: methods, systems and tools. *Communications Surveys & Tutorials, IEEE*, 16(1):303–336, 2014.

[25] Prasanta Gogoi, Monowar H Bhuyan, DK Bhattacharyya, and Jugal K Kalita. Packet and flow based network intrusion dataset. In *Contemporary Computing*, pages 322–334. Springer, 2012.

[26] Stephen Ranshous, Shitian Shen, Danai Koutra, Steve Harenberg, Christos Faloutsos, and Nagiza F Samatova. Anomaly detection in dynamic networks: a survey. *Wiley Interdisciplinary Reviews: Computational Statistics*, 7(3):223–247, 2015.

[27] Cemal Cagatay Bilgin and Bülent Yener. Dynamic network evolution: Models, clustering, anomaly detection. *IEEE Networks*, 2006.

[28] Debajit Sen Sarma and Samar Sen Sarma. A survey on different graph based anomaly detection techniques. *Indian Journal of Science and Technology*, 8(1), 2015.

[29] Danai Koutra, U Kang, Jilles Vreeken, and Christos Faloutsos. Summarizing and understanding large graphs. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 2015.

[30] Misael Mongiovı, Petko Bogdanov, Razvan Ranca, Evangelos E Papalexakis, Christos Faloutsos, and Ambuj K Singh. Netspot: Spotting significant anomalous regions on dynamic networks. In *SIAM International Conference on Data Mining*. SIAM, 2013.

[31] Nicholas A Heard, David J Weston, Kiriaki Platanioti, David J Hand, et al. Bayesian anomaly detection methods for social networks. *The Annals of Applied Statistics*, 4(2):645–662, 2010.

[32] Charu C Aggarwal, Yuchen Zhao, and Philip S Yu. Outlier detection in graph streams. In *Data Engineering (ICDE), 2011 IEEE 27th International Conference on*, pages 399–409. IEEE, 2011.

[33] Carey E Priebe, John M Conroy, David J Marchette, and Youngser Park. Scan statistics on enron graphs. *Computational & Mathematical Organization Theory*, 11(3):229–247, 2005.

[34] Joshua Charles Neil. *Scan Statistics for the Online Discovery of Locally Anomalous Subgraphs*. PhD thesis, The University of New Mexico, Albuquerque, May 2011.

[35] Dan Tang, Kai Chen, XiaoSu Chen, HuiYu Liu, and Xinhua Li. A new detection method based on aewma algorithm for ldos attacks. *Journal of Networks*, 9(11): 2981–2986, 2014.

[36] Brandon Pincombe. Anomaly detection in time series of graphs using arma processes. *ASOR BULLETIN*, 24(4):2, 2005.

[37] Arthur Callado, Carlos Kamienski, Géza Szabó, Balázs Péter Gerö, Judith Kelner, Stênio Fernandes, and Djamel Sadok. A survey on internet traffic identification. *Communications Surveys & Tutorials, IEEE*, 11(3):37–52, 2009.

[38] Charles V Wright, Fabian Monrose, and Gerald M Masson. On inferring application protocol behaviors in encrypted network traffic. *The Journal of Machine Learning Research*, 7:2745–2769, 2006.

[39] Theophilus Benson, Ashok Anand, Aditya Akella, and Ming Zhang. Understanding data center traffic characteristics. *ACM SIGCOMM Computer Communication Review*, 40(1):92–99, 2010.

[40] E Garsva, N Paulauskas, G Grazulevicius, and L Gulbinovic. Packet inter-arrival time distribution in academic computer network. *Elektronika ir Elektrotechnika*, 20 (3):87–90, 2014.

[41] Diane Lambert and Chuanhai Liu. Adaptive thresholds: monitoring streams of network counts. *Journal of the American Statistical Association*, 101(473):78–88, 2006.

[42] Joseph Sexton, Curtis Storlie, and Joshua Neil. Attack chain detection. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 8(5-6):353–363, 2015.

[43] Eric M Hutchins, Michael J Cloppert, and Rohan M Amin. Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. *Leading Issues in Information Warfare & Security Research*, 1:80, 2011.

[44] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography*. CRC Press, 2014.

[45] Rangaswami Balakrishnan and K Ranganathan. *A textbook of graph theory*. Springer Science & Business Media, 2012.

[46] Jagat Narain Kapur. *Maximum-entropy models in science and engineering*. John Wiley & Sons, 1989.

[47] Gregory Reinsel. Introduction to mathematical-statistics, -hogg, rv, craig, at, 1980.

[48] IkamusumeFan. File:gamma distribution cdf.png - wikimedia commons. https://commons.wikimedia.org/wiki/File:Gamma_distribution_cdf.png, Jun 2010. (Visited on 11/21/2015).

[49] Stpasha. File:normal distribution pdf.svg - wikimedia commons. https://commons.wikimedia.org/wiki/File:Normal_Distribution_PDF.svg, Feb 2008. (Visited on 11/21/2015).

[50] Samuel S Wilks. The large-sample distribution of the likelihood ratio for testing composite hypotheses. *The Annals of Mathematical Statistics*, 9(1):60–62, 1938.

[51] Runner1928. Bernoulli distribution chart - bernoulli distribution - wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Bernoulli_distribution#/media/File:Bernoulli_distribution_chart.jpg, Aug 2015. (Visited on 11/21/2015).

[52] Ronald Aylmer Fisher. *Statistical methods for research workers*. Genesis Publishing Pvt Ltd, 1925.

[53] Overview - spark 1.5.2 documentation. https://spark.apache.org/docs/latest/index.html.

[54] Matei Zaharia, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, and Ion Stoica. Spark: cluster computing with working sets. In *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*, volume 10, page 10, 2010.

[55] Benoit Claise, Brian Trammell, and Paul Aitken. Specification of the IP Flow Information Export (IPFIX) protocol for the exchange of flow information. RFC 7011, 2013. URL https://www.ietf.org/rfc/rfc7011.txt.

[56] Brian Trammell, Bernhard Tellenbach, Dominik Schatzmann, and Martin Burkhart. Peeling away timing error in netflow data. In *Passive and Active Measurement*, pages 194–203. Springer, 2011.

[57] Robin Sommer and Anja Feldmann. Netflow: Information loss or win? In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, pages 173–174. ACM, 2002.

[58] Baek-Young Choi and Supratik Bhattacharyya. On the accuracy and overhead of cisco sampled netflow. 2005.

[59] Morton B Brown. 400: A method for combining non-independent, one-sided tests of significance. *Biometrics*, pages 987–992, 1975.

[60] Jim MacLeod. The strange history of port 0. http://www.lovemytool.com/blog/2013/08/the-strange-history-of-port-0-by-jim-macleod.html, Aug 2013. (Visited on 09/22/2015).

[61] IANA. Service name and transport protocol port number registry. https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml, Sept 2015. (Visited on 09/22/2015).

[62] Etienne Stalmans and Barry Irwin. An exploratory framework for extrusion detection.

[63] Wenke Lee and Salvatore J Stolfo. Data mining approaches for intrusion detection. In *Usenix Security*, 1998.

[64] Ibmtivoli application dependency discovery manager. http://ibm.com/tivoli. (Visited on 11/20/2015).