9-16-2011

Scan statistics for the online detection of locally anomalous subgraphs

Joshua Neil

Joshua Charles Neil

Candidate

Mathematics and Statistics

Department

This dissertation is approved, and it is acceptable in quality and form for publication on microfilm:

Approved by the Dissertation Committee:

[illegible], Chairperson

[illegible]

[illegible]

[illegible]

Scan Statistics for the Online Discovery of Locally Anomalous Subgraphs

by

Joshua Charles Neil

B.S., Mathematics, University of California Irvine, 2001
M.S., EE, University of Southern California, 2004

M.S., Statistics, University of New Mexico, 2009

DISSERTATION

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

Statistics

The University of New Mexico

Albuquerque, New Mexico

May, 2011

Dedication

To my wife, Gabby. Her love and patience is always waiting for me at the end of the day. I am truly a lucky man.

To my parents, Charles and Valerie, and my other mothers Jennifer and Camille. They have taught me many things, perhaps most importantly to have love and respect for myself and others.

To my grandparents, Col. Charles V. Neil and Cecil Neil. Their example is one I continually learn from, years after their deaths.

Acknowledgments

As always with work of this kind, it was highly collaborative. It is a plain fact that I have been exceptionally fortunate in the quality of my instructors and colleagues. In other words, I bothered the following people far more than I should have.

First and foremost I want to thank my advisor, Curtis Storlie, who provided a guiding force throughout this research. His technical prowess cannot be overstated, and his patience in instructing me is a credit to him. I imagine I was difficult, at times. He sure was.

Curtis Hash was instrumental in the design and implementation of the path enumeration code. He has created a superb enumeration library, and I am convinced that he will save every one of us. Major data acquisitions were performed by Alexander Brugh, who patiently performed the many queries I asked of him without once complaining about the seemingly fickle nature of my requests. The sheer size of the data sets involved would have prevented me from accomplishing this work without the help of these two excellent computer scientists. My manager, Michael Fisk, and I had many discussions on the behavior of hackers in computer networks. In particular, his insights led to the idea of using paths to identify hacker traversal. In addition, Mike shielded me from the onslaught of tasks a staff member at Los Alamos is normally required to perform, allowing me the time and space to complete this work. Thanks for making things more complicated, Mike!

This acknowledgement would be incomplete without thanking several hundred college professors, so I hereby thank them, very very much. I would like to thank my committee, especially. Professors Curt Storlie, Ronald Christensen and Terran Lane have given me much from their formal instruction, and Scott Vander Wiel continues to provide insight into my current work at LANL.

There are three additional professors in particular that I would like to thank. Professor Hongkai Zhao introduced me to research, and guided me through my first independent work. Professors Mark DeBonis and Mark Finkelstein were the best Algebra and Analysis professors any serious student could hope for.

There are many others. To my family and friends, I hope you all know how much you mean to me!

Scan Statistics for the Online Discovery of Locally Anomalous Subgraphs

by

Joshua Charles Neil

ABSTRACT OF DISSERTATION

Submitted in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy
Statistics

The University of New Mexico

Albuquerque, New Mexico

May, 2011

Scan Statistics for the Online Discovery of Locally Anomalous Subgraphs

by

Joshua Charles Neil

B.S., Mathematics, University of California Irvine, 2001

M.S., EE, University of Southern California, 2004

M.S., Statistics, University of New Mexico, 2009

Ph.D., Statistics, University of New Mexico, 2011

Abstract

Identifying anomalies in computer networks is a challenging and complex problem. Often, anomalies occur in extremely local areas of the network. Locality is complex in this setting, since we have an underlying graph structure. To identify local anomalies, we introduce a scan statistic for data extracted from the edges of a graph over time. In the computer network setting, the data on these edges are multivariate measures of the communications between two distinct machines, over time.

We describe two shapes for capturing locality in the graph: the star and the k-path. While the star shape is not new to the literature, the path shape, when used as a scan window, appears to be novel. Both of these shapes are motivated by hacker behaviors observed in real attacks. A hacker who is using a single central machine to examine other machines creates a star-shaped anomaly on the edges emanating from the central node. Paths represent traversal of a hacker through a network, using a set of machines in sequence.

To identify local anomalies, these shapes are enumerated over the entire graph, over a set of sliding time windows. Local statistics in each window are compared with their historic behavior to capture anomalies within the window. These local statistics are model-based. To capture the communications between computers, we have applied two different models, observed and hidden Markov models, to each edge in the network. These models have been effective in handling various aspects of this type of data, but do not completely describe the data. Therefore, we also present ongoing work in the modeling of host-to-host communications in a computer network.

Data speeds on larger networks require online detection to be nimble. We describe a full anomaly detection system, which has been applied to a corporate sized network and achieves better than real-time analysis speed. We present results on simulated data whose parameters were estimated from real network data. In addition, we present a result from our analysis of a real, corporate-sized network data set. These results are very encouraging, since the detection corresponded to exactly the type of behavior we hope to detect.

Contents

List of Figures

List of Tables

Glossary

γ	A window in the time $\times$ graph cross product space
λ_γ	The glrt statistic calculated on an observed window γ
λ_e	The glrt statistic calculated on a single edge e
A^T	The transpose of a matrix or vector A

Chapter 1

Introduction

In this dissertation, we consider the problem of detecting locally anomalous activity in a set of time-dependent data having an underlying graph structure. Our motivational data set, a large computer network, yields graphs on the order of 20 thousand nodes and 500 hundred thousand edges, every 30 minutes. We describe a method to identify, in real-time, attacks occurring in this network. In particular, we are interested in identifying local regions within the graph that have deviated from historic behavior in some window of time.

1.1 Motivating Example

To help clarify the idea, we first describe a scenario that we are interested in detecting. Consider an attack by a hacker on a computer network. A common initial stage of the attack is to infect a machine on the network, using malicious software. One approach is to use a phishing attack. An example of a phishing attack is an email which includes a link to a malicious website. When the user clicks on the link, their machine becomes infected, giving the attacker some form of access to the machine. The attacker generally cannot dictate which machine is infected, and the initial host is usually not the ultimate target of the attack, if there even is an ultimate target. Therefore, from this initial host, the attacker may proceed to other hosts, hopping from one to the next. See Figure 3.1 for a visualization of this traversal.

Once the hacker has gained an initial foothold, he might then proceed to more important network assets from this initial host. The target may be a database machine with important data, or an administrative machine with the credentials to access many hosts on the network. Another goal is to simply obtain access to as many accounts as possible on the network.

In any case, as the attacker traverses the network, he create anomalous activity in the time series along each edge he traverse. The union of these edges yields an anomalous shape in the network graph, in a given period of time. It is these shapes which we seek to detect.

1.2 General Approach and Organization of This Work

The goal of this approach is to discover, for a predefined shape, the *most anomalous of these shapes in the graph* over a set of time windows defined on each edge. We are interested in testing the null hypothesis that all edges in the graph are behaving as they have historically, versus alternatives indicating that there are local clusters of altered activity among the edges connected in some local shape in the graph. To accomplish this goal, we have developed a method based on scan statistics to examine each of these shapes in the graph, over sliding windows of time.

Scan statistics have been widely used to detect local clusters of events [10, 15, 24, 19]. The idea is to slide a window over a period of time and/or space, calculating a local deviation statistic. The maximum of these is known as the scan statistic. Under a null hypothesis of homogeneity, an observation is compared to the distribution of the scan statistic to obtain a p-value, which measures deviation from historic behavior in the local window.

In this work, we focus on two particular shapes in the graph: stars and paths. While the use of stars as a scan object has been examined [28], the path scan object is novel to the literature. We make the case for using paths to discover anomalous traversal both in simulated examples (see Chapter 7) and in a real world network example (see Chapter 8), using data extracted from the internal network at Los Alamos National Laboratory (LANL).

Both path and star anomalies have been observed in attacks in LANL's computer network. A star anomaly (see Section 3.2) is indicative of a hacker using a compromised machine to attempt to connect to every machine it can reach, creating anomalies on every edge emanating from the compromised host. A path anomaly (see Section 3.1) indicates a more subtle attack, which, at its core, is a sequence of traversals from each host in the path to the next host. We also discuss an anomalous shape that is a mixture of stars and paths, the caterpillar, in Section 6.5.

To enable us to identify when current behavior has deviated from historic behavior, we require models of historic behavior. Therefore, we develop and estimate models on data arising from more than 250,000 unique edges in a real-world network data set. See Section 4.3 for data collection and preprocessing issues, Section 4.2 for the models used in simulation, and Section 9.1 for ongoing work on modeling this type of data.

We then study various simulations using these estimates. We present results when path anomalies are present in Section 7.1 and when other types of anomalies are present in Section 7.2. In addition, we present results of scanning on real computer network data in Chapter 8.

This approach was designed to monitor a computer network in real time, and any scheme applied to computer network data at an enterprise-level (20,000 or more individual IP addresses) needs to be fast. Yet, in order to identify highly local anomalies, the system needs to monitor many small windows simultaneously. We have designed and implemented a prototype monitoring system which is capable of examining a large number of extremely local objects in a corporate-sized network in real-time. The components of this system are described throughout this work, but the core algorithm and its performance are described in Section 3.1.1.

1.3 Prior Anomaly Detection Systems at LANL

Anomaly detection has been an important part of LANL's cyber security efforts for a relatively long time. LANL is an attractive target for hackers for two main reasons. First, we are a large network, with all of the computing power that this size implies. This is useful to a hacker in its own right, since resources can be used for other nefarious activity, such as using the compromised machines for the sending of spam. The second reason is perhaps more compelling: we are a keeper of the nation's secrets, and, as such, we represent a high priority for some.

In addition, LANL has the advantage of being able to extract large amounts of data from the network. We have archival data sets reaching back to the late 90's, which, while they are invariably incomplete, do represent a massive amount of historical network data. We have near real-time collection taps, collecting terabytes of data per day.

In the following subsections, we describe two detection systems which influenced our current work. Both of these systems monitor data extracted from the time-series of communications on the network. Neither, however, is designed to capture locality in the graph, as this current work does.

1.3.1 EMAAD

The first system is called the Exponential Moving Average Anomaly Detector (EMAAD). EMAAD was designed and implemented in 2005, and has been running as an online system at LANL since then. This system uses an exponentially weighted moving average (EWMA) to maintain a mean and variance estimate of the current data stream. This is a simple approach to handling time-series data, and its computational speed is impressive. But the cost of this simplicity is the inability to capture less smooth types of behavior. For example, the data may have periodic spikes, and an anomaly detector needs to be able to handle these periodic spikes. Heterogeneous models could be used to handle this pattern, but EMAAD was built to keep very little state, and so heterogeneous models were not used. Instead, the system uses an asymmetric moving average. The exponential weight is smaller when the observed

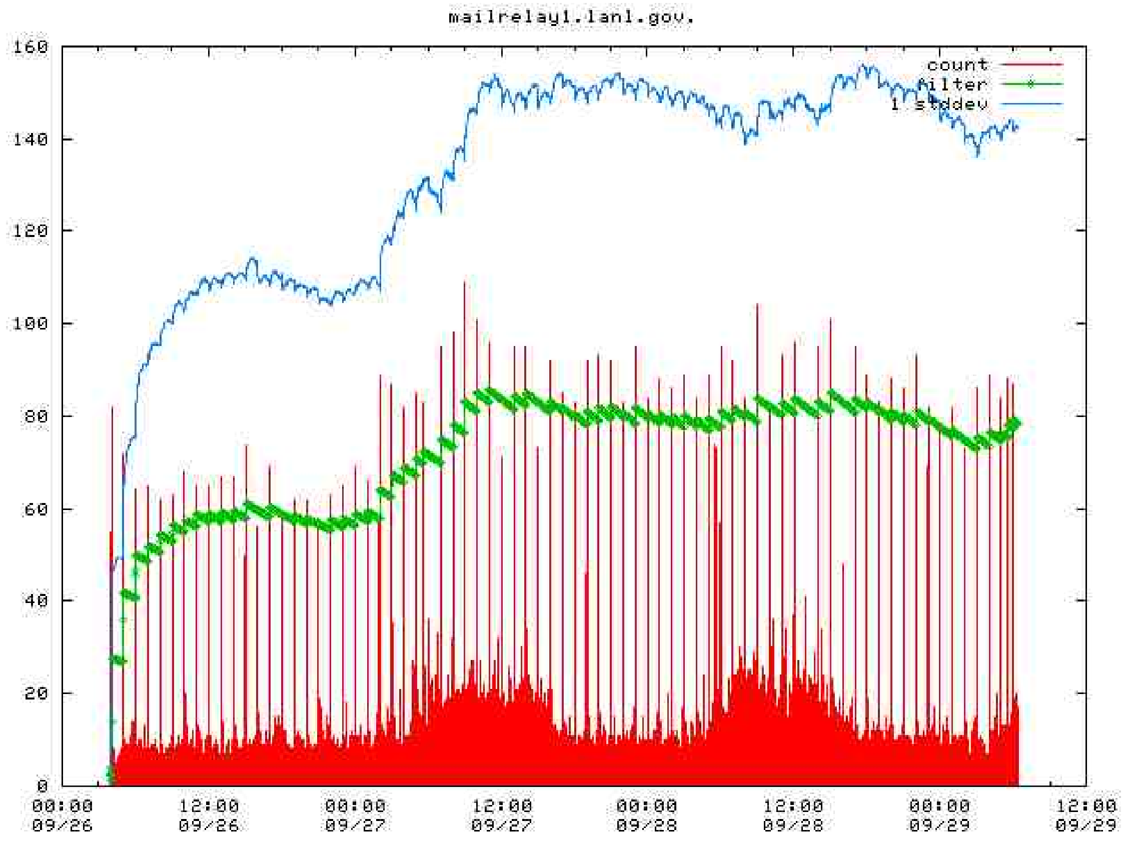


Figure 1.1: EMAAD System Performance

data is larger than the current mean estimate, allowing the mean estimate to adjust quickly to these spikes. In order to avoid alarming on future spikes, the system has a much smaller EWMA weight when the data is smaller than the mean, leading to a slower adjustment to lower counts. This allows a rapid increase in the model, at the cost of poor fit after the rapid increase, which results in insensitivity between spikes. Figure 1.1 shows system performance on spikey data. Observe that the mean estimate is near the level of the spikes, and therefore any anomaly below the level of the spikes will not be captured.

1.3.2 KATS

To alleviate this insensitivity, our team designed and implemented a second generation detection system, known as Kernel-Smoothed Adaptive Thresholds (KATS). This system is based on the method described in [16], but with significant modifications. The basic approach is to define a grid of parameter estimates over the day. This grid can be defined at whatever time resolution is desired, and currently uses a ten-minute spacing. Parameters for a given time point are found using locally weighted regression on the grid, to allow for spikes and other rapid changes over time. To handle day-of-week effects, KATS models each day separately, and the previous similar day's estimates are updated with today's estimates using an EWMA. For example, this Monday's grid estimates are EWMA'd with last Monday's estimates to provide next Monday's predictive parameters.

Finally, to get an anomaly score at time t, we use the parameters at time t to define a Negative Binomial distribution. Since we are concerned with anomalously high counts C_t, indicating hacker activity, the upper p-value, ρ_t, is calculated for the observed count c_t, according to this distribution. In other words,

$$\rho_t = P(C_t > c_t) = 1 - F_{NB}(c_t|\mu_t, \sigma_t^2)$$

where $F_{NB}(c_t|\mu_t, \sigma_t^2)$ is the negative binomial distribution function with mean (at time t) given by μ_t and variance given by σ_t^2. The normal score, $Z_t = \Phi^{-1}(\rho_t)$ is then evaluated. High count anomalies, which are the only kind we are interested in, will correspond to negative Z_t, and, if the data were generated from the KATS model, they would be normally distributed. Instead of using these directly, however, we pass the Z_t to a CUSUM control chart. $C_t = \min\{0, C_{t-1} + Z_t - \omega\}$, where $\omega > 0$ is an overall tuning parameter. This allows for alarming on accumulations of moderately anomalous (negative) values over time, or alarming on a severe instantaneous anomaly. See [31] for a good reference on CUSUMs. A threshold for C_t is set using historic data to alarm at a given rate.

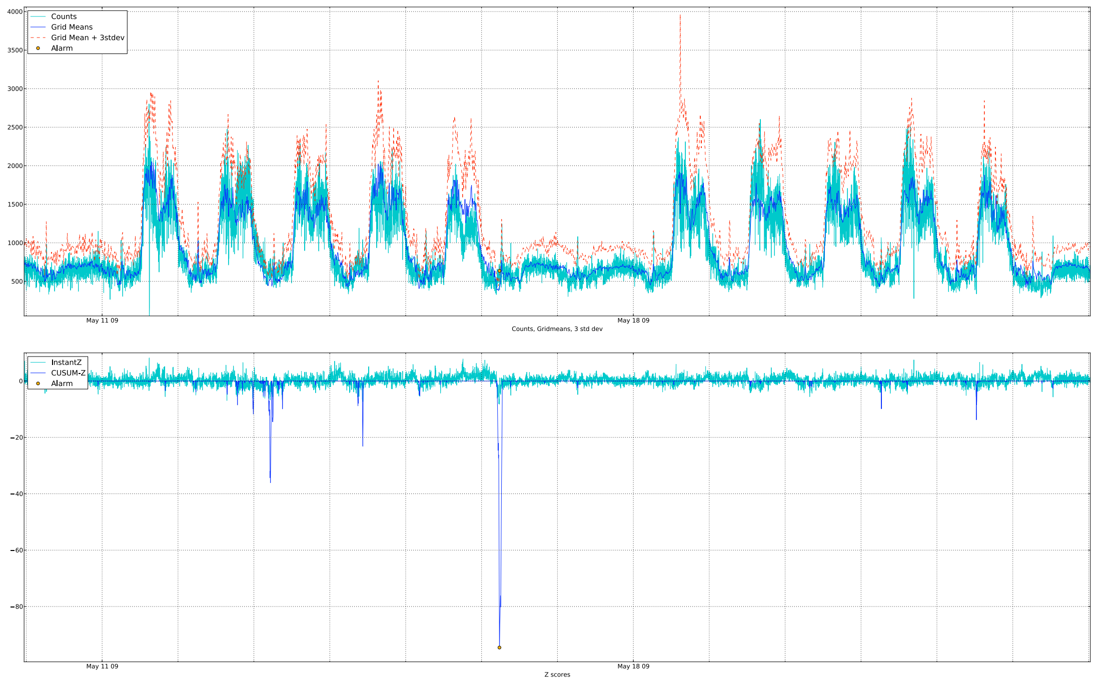


Figure 1.2: KATS System Performance

System performance can be seen in Figure 1.2. The upper plot provides the data, mean estimate, and standard deviation estimate. The lower plot gives the instantaneous and CUSUM'd anomaly scores.

1.4 Related Work

Fast statistical anomaly detection on streaming data has become an important area of research, given the proliferation of data over the past few decades, and the need to detect quickly the event that a process has changed significantly from past behavior. Applications can be found in many areas including engineering [5], computer science [9, 18], and, specifically, in communications networks [23, 16, 33, 5].

In many cases, the data can be represented as a graph [14]. Nodes represent actors sending and receiving data, and edges represent communications between nodes. To take time into consideration, we have several options. For example, graphs could be considered as dynamic. In this paradigm, edges appear and disappear according to whether or not there is active communication between the two nodes. Anomalies can be detected in the changes to the structure of the graph [28, 6, 25]. These methods tend to lack fine-grained locality, something we address specifically, using path scans, in Section 3.1.

Another approach is to define a fixed graph based on historical communications. Changes over time can be captured by time series representations on each edge. These time series can then be monitored for change. A simple approach would be to aggregate all communications emanating from each node, and consider nodes to be independent [33, 23]. In many networks, however, node behavior may not be independent, and different edges may have significantly different behavior over time, so that modeling each edge is desirable. In addition, traversal cannot be captured by only analyzing node behavior.

In [11], individual edges are modeled, and a Bayesian testing framework is proposed to test the anomalousness of each edge, without consideration of other local edge anomalousness. These edges are then passed to a secondary analysis, which looks for centrality in the graph constructed from the edges which were detected in the initial pass. Centrality in the anomalous edge graph can be detected in this way, but simultaneously testing multiple local sets of edges will have increased power to detect locally anomalous behavior. For example, if two anomalous edges were connected by a non-anomalous edge, this possible traversal path would likely be missed by the technique in [11], but is a valid anomaly in many settings. In addition, when data speeds are high, a fully Bayesian treatment may not be feasible computationally, unless the model is parsimonious enough for sequential Monte Carlo [7].

Scan statistics have been used to detect anomalies in the Enron email graphs [28]. While they scan for anomalies in the out degree of every node in the graph over time, our method examines any general shape the user wishes to define. In Chapter 6, we discuss two shapes, paths and stars. Stars are similar to the data that the EMAAD system monitors (see Section 1.3.1), but EMAAD aggregates every out edge into a single data stream, while star scanning models each out edge individually. We show that paths capture very general anomalous shapes, but that star anomalies are too global to identify anomalous paths. Star-shaped windows, however, are still important, since they encompass a type of anomaly seen in attacks in the past, and, as seen in Section 7.2.1, capture star anomalies well.

In addition, anomalous paths have been examined [20] using a similarity metric to compare paths, and clustering to find outlying paths. This method, however, assumes we observe a "path value", that can be clustered upon. On the contrary, in this work we propose a statistically rigorous method to infer anomalous shapes from the network without any prior knowledge about traversal by individual actors.

One common thread to most of the work mentioned above is that it is appropriate for data on much smaller scales than our method proposes to address. We have a data set that is difficult to come by, as it consists of a record of most of the communications between individual computers on a large corporate-sized network. This data is recorded every minute, and has been archived for the past decade in some cases. The objects we monitor number in the hundreds of millions per 30 minute window, and we are able to test 30 minute windows in under 5 seconds, using a compute machine with 48 cores.

The sheer size of this endeavor, we believe, separates it from most other work on graph-based scan statistics, and anomaly detection in general. The telephone network literature [16, 17], monitors larger networks than ours, but not in a graph based setting, instead monitoring aggregation points in the network. The graph enumeration engine at the core of the system scales extremely well. In Chapter 3, we describe the algorithm, and briefly discuss performance on several large graphs.

Chapter 2

Methodology

In this chapter, we describe the methodology behind scanning for local anomalies in a graph over time. We discuss windowing in this space, followed by the definition of the scan statistic. Finally, we discuss issues involving the comparison of different sized windows.

2.1 Windows in the Cross Product Space

We are interested in examining sets of windows in the $Time \times Graph$ product space. We define these sets of windows as follows. We have a graph $G = (V, E)$ with node set V and edge set E. For each edge $e \in E$, at discrete time points $t \in \{0, \ldots, T\}$, we have a data process $X_e(t)$. We denote the set of windows of time on edges e over discretized time intervals $(s, s+1, \ldots, k)$ as

$$\Omega = \{[e, (s, s+1, \ldots, k)] : e \in E, 0 \leq s < k \leq T\}.$$

The set of all subsets of windows, $\Gamma = \{\{w_1, w_2, \ldots\} : w_j \in \Omega\}$ is usually very large, and we are normally interested in only a subset $\Gamma_s \subset \Gamma$ which contains locality constraints in time and over the graph. We therefore restrict our attention to sets of windows $\gamma \in \Gamma_s$. We note that Γ_s is usually problem dependent, and we give specific examples in Chapter 3. For convenience, we denote $\boldsymbol{X}(\gamma)$ as the data in the window given by γ.

Next, we assume that for any time point t and edge e, we can describe $X_e(t)$ with a stochastic process with parameter function given by $\theta_e(t)$. By $\boldsymbol{\theta}(\gamma)$, we denote the values of the parameter functions evaluated in the corresponding set of windows γ . Finally, the likelihood of the stochastic process on γ is denoted $\mathcal{L}(\boldsymbol{\theta}(\gamma)|\boldsymbol{X}(\gamma))$. We give specific examples of likelihoods in Chapter 4.

2.2 A Scan Statistic for Windows in the Cross Product Space

We would like to examine whether the data in a window could have been produced by some known function of the parameters $\hat{\boldsymbol{\theta}}(\gamma)$, versus alternatives indicating that the parameters have changed. That is, given that we observe $\boldsymbol{X}(\gamma) = \boldsymbol{x}(\gamma)$, we would like to test whether $H_0 : \boldsymbol{\theta}(\gamma) = \hat{\boldsymbol{\theta}}(\gamma)$ against alternatives that can be formed by restricting the overall parameter space, $\boldsymbol{\Theta}$, to a subset $\boldsymbol{\Theta}_A \subset \boldsymbol{\Theta}$. The generalized likelihood ratio test statistic (glrt) is a natural statistic to use. Let

$$\lambda_\gamma = -2\log\left(\frac{\mathcal{L}\left(\hat{\boldsymbol{\theta}}(\gamma)|\boldsymbol{x}(\gamma)\right)}{\sup_{\boldsymbol{\theta}\in\boldsymbol{\Theta}_A}\mathcal{L}\left(\boldsymbol{\theta}(\gamma)|\boldsymbol{x}(\gamma)\right)}\right) \tag{2.1}$$

While λ_γ is a valid measure of anomalousness, it depends on the number of parameters being tested in the window. In the next section and in Chapter 5, we discuss removing this dependence by estimating the distribution of λ_γ to obtain p-values.

2.3 Comparing Variable Sized Windows

Large values of λ_γ give evidence against the null, so we could use λ_γ to quantify deviations of the data from historic behavior. The distribution of this statistic, however, is dependent on the size of the window, and we wish to compare windows of different sizes. To remove this dependence, we calculate p-values, i.e. $\mathcal{P}\left(\Lambda_\gamma > \lambda_\gamma\right)$, where Λ_γ has the distribution of the test statistic under the null hypothesis, and γ denotes the window in question, and this distribution is needed for all $\gamma \in \Gamma_s$. Since there may be many windows γ, this can be a challenging task.

For simple tests, this distribution may have a known form, but for more complicated scenarios, monte-carlo samples can be taken to approximate it. This approach can be problematic, however. When the number of objects tested is large, we obtain, due to normal variation, p-values which are beyond machine precision. Alternatively, we can use the classical asymptotic approximation of $\Lambda_\gamma \sim \chi^2_\nu$, where ν is the number of parameters being tested.

However, this asymptotic result depends upon the null parameters being in the interior of the alternative parameter space. If, for example, we are testing for an increase in a parameter, then the historic parameter will be on the boundary of the space, and the χ^2 approximation is not valid. In fact, in this case the distribution will have a point mass at 0, since roughly half of the density will be for values of the parameter estimated to be less than the historic parameter, under the null. See [4], (pg. 516). Specifically, it is required that $\hat{\theta}$ be in an open subset of the alternative parameter space. The setting is even more complex when more than one parameter is being tested simultaneously. We present two approaches to calculating the distribution of Λ_γ, which alleviate these issues, in Chapter 5. In any case, once approximate p-values are calculated, say p_γ, their marginal distribution, under the null hypothesis, is approximately Uniform$(0, 1)$ and hence does not depend on the window size under consideration.

To scan for anomalies in the (*graph* $\times$ *time*) product space, we must slide over all windows γ, keeping track of the scan statistic

$$\Psi = \min_\gamma p_\gamma$$

In practice, thresholding must be done on the set of p-values, and so more than just the minimum p-value should be considered. For online monitoring, we set a threshold on the p-values to control the false discovery rate [1]. We describe this threshold setting in more detail in Section 5.4, but we emphasize that when a detection occurs, a set of windows are detected, and so the union of these windows is the detected anomaly produced by the system.

To this point, we have been very general in our description of Γ_s, the set of sets of windows to be scanned. In the next chapter, we describe more specific window shapes for detecting anomalous computer network behavior.

Chapter 3

Local Windows in the Graph

The approach described in Section 2.2 can be used for batch (retrospective) or online (prospective) processing. If a system is working in an online setting, it might perform some automated action, such as quarantining a host on a computer network, or sending these alerts to an analyst for further, offline analysis. But graphs are combinatorial in nature. For a fully connected graph with n nodes, the number of subgraphs is $2^{n(n-1)}$. A fully connected graph with 17 nodes has more subgraphs than the number of atoms in the observable universe. This motivates the need for an extremely restricted set of graph windows, especially in the online setting.

3.1 Directed k-Paths

In addition, we wish to construct windows which are appropriate for identifying specific shapes of anomalies. Since a primary motivating example is the example of hacker traversal in a computer network, we suggest a specific type of subgraph for

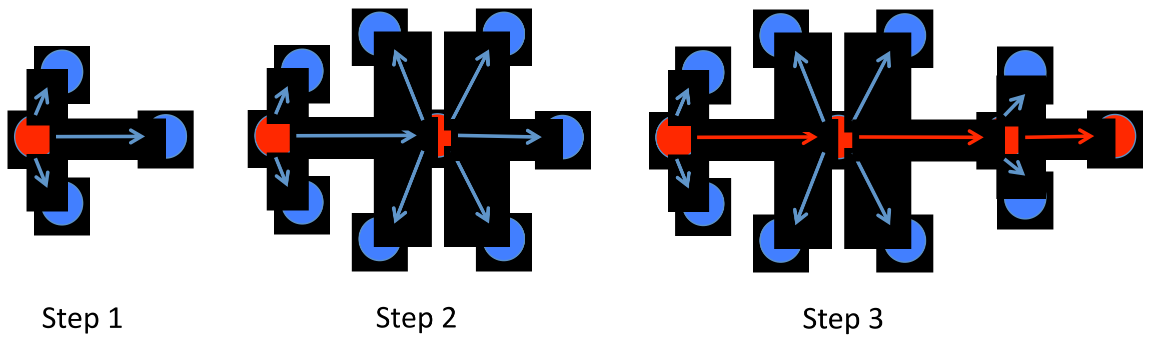


Figure 3.1: Steps in a Traversal Attack. Step 1: Initial infection and local search. Step 2: First traversal has occurred, and further search is performed.
Step 3: A full traversal has occurred. We denote this shape as a Caterpillar. The Path at the core of the anomalous shape is highlighted in red, but the data on each edge in this graph is potentially anomalous.

online monitoring: directed k-*paths*. A directed k-path is a set of edges of a subgraph of size k, which has diameter k. Here, size is the number of edges in a graph, and diameter is the greatest distance between any pair of nodes. Informally, this just means that a k-path is a sequence of edges where the destination node of the current edge in the sequence is the source node of the next edge in the sequence, and so on.

In terms of computer networks, this reflects the traversal of an intruder who makes a set of anomalous edges, moves to one of the newly infected hosts, and repeats the process from that host. The k-path has the advantage that it captures the core of many network attacks, since the attack is described by a path through the network, with additional edges as "fuzz" around this core path. See Figure 3.1. We note that this attack shape has been observed in actual attacks on LANL's network, and is therefore an important focus of this work.

For the simulation and real data studies described in Chapters 6 and 8 respectively, we choose to use 3-paths. 3-paths have the advantage of locality, but are also large enough to capture significant traversal. In order to scan every 3-path in the network graph, we must first enumerate every 3-path. This can be non-trivial for many graphs. In a fully connected graph with n nodes, and eliminating cycles and back edges, we have $n(n-1)(n-2)(n-3)$ 3-paths.

In reality, our network graph is much less connected. However, in a thirty-minute window of time, if we only include edges with non-zero activity in that window, we obtain a graph which is around 17,000 nodes, 90,000 edges, and approximately 300 million 3-paths. We note that, while we effectively scan this entire set of $n(n-1)(n-2)(n-3)$ 3-paths, we do not calculate an anomaly measure on any path with an edge that has no activity in the current time window. Since a hacker needs to make at least one communication per edge to traverse this edge, no activity on an edge indicates no traversal over that edge, and this path is therefore not considered anomalous.

3.1.1 Path Enumeration Algorithm

Due to the large number of 3-paths, it is highly important to be able to enumerate paths quickly, if we hope to maintain a near real-time response time. We have written a fast, parallel C library for this purpose. At the core of the library is an algorithm which enumerates paths, as described in Figure 3.2. We note that this algorithm takes very little memory, and is trivially parallelizable.

We tested the performance of the enumeration algorithm on several graphs, without adding the additional calculations required to test anomalousness, in order to get an idea of its raw performance. To get a feel for the system performance on a large portion of LANL's network, we added the vulnerability scanning machines

```
Type definition:
    An edge A is a list of length 2, where A[1] is the source node and A[2] is
    the destination node

function ENUMERATE(E, K):
    // E = the list of edges representing a graph
    // K = the integer length of paths to enumerate
    for each edge A in E: // A is some edge in the graph
        list P[1] = A // A becomes the first edge in a path
        RECURSE(E, P, 1, K) // recursively append additional edges

function RECURSE(E, P, L, K):
    // E = the list of edges representing a graph
    // P = the list of edges representing a path
    // L = the integer length of P
    // K = the integer length of paths to enumerate
    edge A = P[L] // A is the last edge in the path
    for each edge B in E: // B is some edge in the graph
        if A[2] == B[1] then:
            P[L+1] = B // B becomes the last edge in the path
        if L+1 == K:
            EMIT(P) // a K-path was found
        else:
            RECURSE(E, P, L+1, K) // recursively append additional edges
```

Figure 3.2: Path Enumeration Algorithm

discussed in Section 4.3.1, yielding a graph with 65,536 nodes and 12.7 million edges. The algorithm enumerated 30.4 trillion 3-paths in 2.2 hours on a 200-core commodity cluster. In a second example, the method was run on an a data set taken from the Stanford SNAP Graph Library Data Sets [30]. Running on the LiveJournal (an online social network) data set, resulted in a graph with 4.8 million nodes and 69 million edges. This led to 825 billion 3-paths, enumerated in 3.3 minutes.

While these numbers are interesting for general path enumeration, our focus, of course, is on the computation cost when evaluating path likelihoods and testing for anomalies, in addition to enumerating the paths. In the simulations discussed in

Chapter 6 and the real data example in Chapter 8, we were able to enumerate and test 30 minute windows consisting of roughly 300 million paths, in under 5 seconds per window. This allows us room to add complexity to the models, and handle larger graphs than the already sizable graphs we are currently analyzing, while keeping up with real-time data streams.

3.2 Stars

We now examine another set of shapes: stars. Specifically, we are using stars defined as the set of edges whose source is a given central node. See Figure 3.3. While these shapes are not very localized, especially for high out-degree nodes, they still pick up star-type anomalies rather well. This shape is important, since it is a shape resulting from a hacker examining all out-edges of a given compromised host, and is an attack behavior we have observed in practice against LANL's computer networks.

Current systems employed at the laboratory (See Section 1.3) model the aggregation of these edges to detect star-shaped attacks, but modeling each edge individually gives us much greater power over aggregation, since individual edge anomalies can provide greater fidelity to attacks occurring on only a subset of the out-edges.

We will see that paths have the ability to describe a more rich class of shapes than star windows, but star windows generally outperform paths on star anomalies. While star anomalies are created by perhaps less sophisticated adversaries, we should not ignore this class just based upon lack of sophistication.

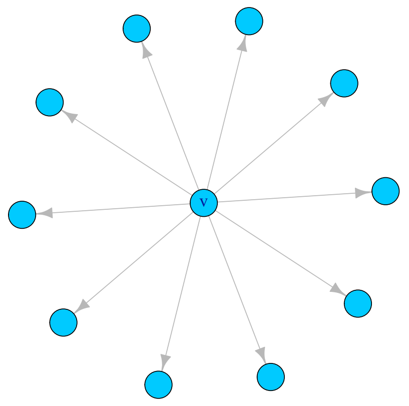


Figure 3.3: Example Out-Star, centered at node v.

3.3 Windows in Time

Now that we have discussed the shapes of windows in the graph, we briefly discuss shapes in time. The simplest shape is the same interval of time over every edge in the graph window. This will detect anomalies which occur in the same window for each edge in the shape.

For paths, we can discuss more sophisticated time-windows. For example, in a directed path, sequentially incremented time-windows over the sequence of edges would capture anomalies occurring sequentially. Telescoping windows are another option which may capture this nested anomalous activity. For example, if one opens an interactive shell on a remote machine via the ssh protocol, and then from that open connection initiates a further connection, then the first connection encompasses the second in time, and a telescoping window would capture this activity. In order to keep things more general, and manageable, however, for the simulations and real data examples that follow, we consider windows of time that are the same across all edges.

Chapter 4

Edge Models For Evaluation

4.1 Overview

To scan for anomalous shapes, it is necessary to have models for each edge, and parameter estimates for each of these models. In this work, we use two distinct models on edges, both of which are motivated by the observation that it is common in computer network data to observe a switching process. For an example of this switching behavior, see Figure 4.1. Intuitively, for many edges this is caused by the

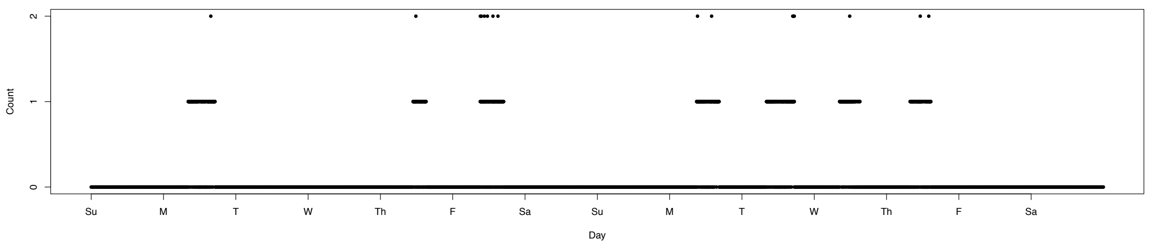


Figure 4.1: Plot of Example Edge Data. The plotted value is number of Netflow (TCP) connections per minute. The flows originate at a specific user's machine, and the destination is a server providing virtual desktop services.

human presence on the network. If a user is present at a machine, he will make non-zero counts on edges emanating from that machine. But users are not computing 24 hours a day. Coffee breaks are mandatory! This presence/absence induces a switching process between a purely 0 count emission and a higher activity count emission. While intuitively, there will be higher counts in the middle of the day than at night, in this work we use homogeneous models for the sake of simplicity. Our simulation models were chosen to somewhat reflect the data, but the focus here is on the method of scanning, and not on modeling of edges. See Chapter 9.1 for a discussion on other possible models, which explicitly account for diurnal behavior.

4.2 Model Descriptions

The first model is a *two-state observed Markov model* (OMM), which we denote B_t. If there was a non-zero count in time bin t, then $B_t = 1$, otherwise, $B_t = 0$. This model has two parameters,

$$p_{01} = P\left(B_t = 1 | B_{t-1} = 0\right) \quad \text{and} \quad p_{10} = P\left(B_t = 0 | B_{t-1} = 1\right).$$

The second model is a *two-state hidden Markov model* (HMM) [29], with a degenerate distribution at zero for the low state, and a negative binomial emission density in the high state. Negative binomial emission densities do not suffer from the equidispersion property of the Poisson [27], and a good justification for using them to monitor for anomalies in network counts is given in [16]. We note that this model is similar to the hurdle models described in [11] and elsewhere, with one important distinction: we allow the high-state to emit zeros. This complicates estimation, since it introduces a latent variable. However, we believe that this is important in modeling our data. Again, referring to Figure 4.1, we see that zero counts are interspersed with the non-zero data, but are still clearly a part of the 'active' state. Intuitively, we think of the active state as "The user is present at the machine", and therefore likely to make communications, not as "The user is making a communication on this edge".

The HMM, call it H_t, has a "hidden" Markov process, call it Z_t, which makes transitions between 0 and 1. The transition parameters are given by

$$p_{01} = P\left(Z_t = 1 | Z_{t-1} = 0\right) \quad \text{and} \quad p_{10} = P\left(Z_t = 0 | Z_{t-1} = 1\right).$$

We parameterize the emission densities in both states as

$$\begin{aligned} P(H_t|\mu, s, Z_t = 0) &= I(H_t = 0) \\ P(H_t|\mu, s, Z_t = 1) &= NB(H_t|\mu, s) \end{aligned}$$

where $I(\cdot)$ is the indicator function and $NB(\cdot|\mu, s)$ is the Negative Binomial density function with mean μ and size s.

To obtain likelihoods for paths, we assume independence across the edges. Under this assumption, Equation 2.1 becomes

$$\lambda_\gamma = \sum_{e \in \gamma} \lambda_e \tag{4.1}$$

where λ_e are the glrt scores on each edge in window γ. This means that edge glrt scores can be calculated before path enumeration, and the per-path calculation is minimized. Since the number of paths is usually many orders of magnitude larger than the number of edges, limiting the per-path computational complexity is of paramount importance.

Certainly, with the number of paths in question, modeling general dependence among edges would be computationally difficult, and assuming independence is a reasonable compromise given the computational complexity of modeling general dependence, and the sparseness of the data. A major exception is for bi-directional edges, the set of two edges such that the first edge's source is the second edge's destination, and the first edge's destination is the second edge's source. These edges, in computer networks, do, in fact, potentially reflect a correlated set of activity between two machines, since, for some protocols, this behavior is standard, and represents a single bi-directional stream of flows. For this work, we do not include bi-directional edges in a path, so that the likelihoods on each edge in a bi-directional edge are never added together. In future work, we intend to aggregate bi-directional edges for the purposes of modeling and testing, but not for window enumeration.

4.3 Data Collection and Preprocessing

To define an overall graph and collect edge data on the graph, we use flow data [26, 21, 3] gathered from the communications on one of LANL's internal networks, over thirty days, starting January 30, 2010. IP addresses define nodes, and any directional flow between IPs defines the existence of a directed edge in the graph. To further simplify the data, we focus only on TCP connections, excluding other layer 4 protocols, since the majority of the communications are captured by TCP. We note that it is not necessary to eliminate other layer 4 protocols, but, for initial development, it was useful, in that it reduced the computational load.

4.3.1 Vulnerability Scanning Machines

In addition, we removed a subnet of machines which are involved in vulnerability scanning of the network. Vulnerability scanning is an automated process that scans each host in the network, looking for potential holes in the security of the hosts. Since these machines represent normal activity on the network which is not malicious, but generally create highly connected graphs if they are not removed, including creating edges to non-existent hosts on the network, it was deemed appropriate to remove them. In fact, removing these nodes and their edges reduced the number of 3-paths from 30 trillion to approximately 1.5 billion, for the thirty days in question. This results in a graph with 20,382 nodes and 558,785 edges. See Figure 4.2 for a visualization of this 30-day graph.

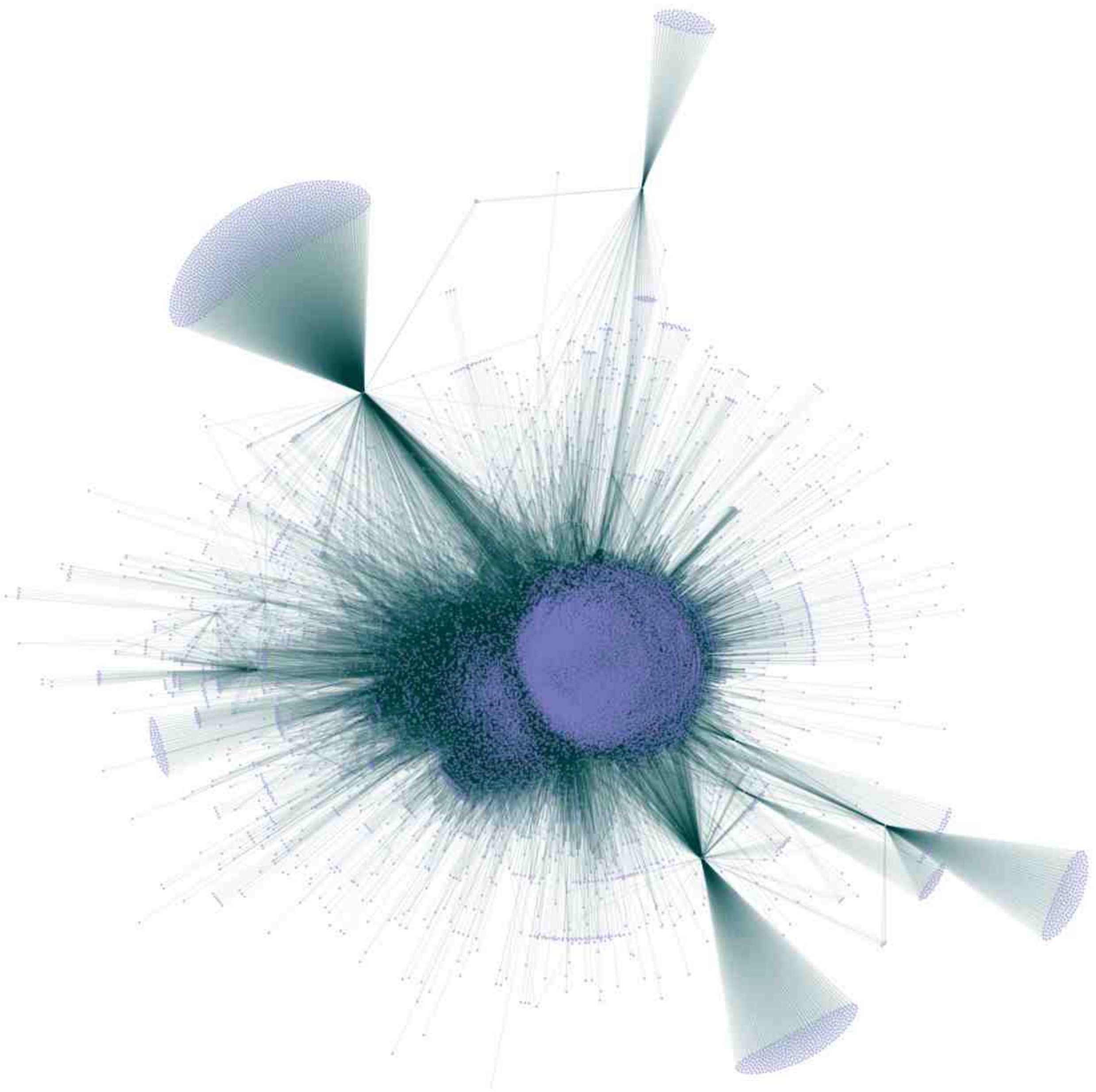


Figure 4.2: LANL Network Graph Over 30 Days

4.4 Model Estimation

We require parameter estimates for each of the 558,785 edges. But many of these edges are very sparse, and, therefore, there is not much opportunity to observe high state counts. To perform the estimation, we pool edges according to μ_e, the average number of non-zero counts per day, averaged over all 30 days. We define two Edge Types:

- *Edge Type I* ($\mu_e \geq 1$) consists of those 252,165 edges (45%) for which we have sufficient data to estimate an individual model. We estimate the parameters on each of these edges by maximizing the likelihood.
- *Edge Type II* ($\mu_e < 1$) consists of the remaining 306,620 edges (55%). These edges will share a common parameter set, in order to borrow information across very sparse data. To avoid becoming overly sensitive on edges in this set, we extract the set of edges $\tilde{e}$ such that $\mu_{\tilde{e}}$ was among the 1,000 largest μ_e values in Edge Type II. We then estimate parameters on each of these edges, and take the mean of these parameter vectors. The common edge model for Edge Type II is then parameterized by this vector. Since our tests are focused on increased activity, taking the largest 1,000 μ_e will ensure that the models are not overly sensitive on these low count edges.

For the OMM, maximum likelihood estimation is straightforward. For the HMM model, an expectation-maximization approach (see [2]) was used to obtain maximum likelihood estimates. Estimation on 252,165 edges was somewhat intensive, taking approximately 1.48 cpu years to complete on a 256 node cluster. However, the full parameter estimation scheme does not need to be performed/updated at the time scale of the scan windows in practice. Rather, this updating can be done nightly or weekly on new data. For more details on updating, and a more sophisticated approach to ‘borrowing’ strength across edges, see Section 9.1.7.

Chapter 5

P-value Calculation and Threshold Determination

We seek a p-value for the observed glrt statistic, λ_γ. Under mild conditions, the generalized likelihood ratio test statistic is asymptotically χ^2 with degrees of freedom equal to the number of free parameters in Θ. This holds only when the true parameters are not on the boundary of Θ, however. If the true parameters are on the boundary, we will obtain a point mass at zero in the distribution of λ_γ. This is precisely the case in the testing we present in Chapters 6 and 8, and results from the fact that the types of activity we are looking for in the computer network are tested by looking for an increase, but not for a decrease, in certain parameters in the model.

Therefore, we must rely on other methods to determine p-values for λ_γ. We will proceed by establishing a model for λ_γ in each of the two scan shapes, star and path. Recall from Equation 4.1 that under independence of the individual edge distributions, the glrt score for window γ is the sum of the glrt scores on each edge e in γ. Since it is possible that for every edge $e \in \gamma$, λ_e is zero due to restrictions on the parameter space, we must allow a point mass at zero in the distribution of λ_γ.

5.1 Star p-values

We start with the simpler of the two shapes, the star. The number of stars in a graph is just the number of nodes, and therefore, for each node v, we can model the distribution of

$$\lambda_v = \sum_{e \in outedges(v)} \lambda_e$$

for the star around v.

5.1.1 Modeling

Let Λ_v have the distribution of the λ_v. We model Λ_v as

$$\Lambda_v = B_v X_v$$

where $B_v \sim \text{Bernoulli}(p_v)$ and $X_v \sim \text{Gamma}(\tau_v, \eta_v)$.

Since all λ_e in the sum could be zero, Λ_v must have a point mass at zero. This is captured by B_v. To model the positive part of the distribution for Λ_v, the Gamma distribution is attractive since it is equal to a χ^2 distribution with degrees of freedom ν when $\tau_v = \frac{\nu}{2}$ and $\eta_v = 2$. In this way, we allow the unrestricted asymptotic case as one model choice.

5.1.2 Estimation

To estimate τ_v and η_v, we use direct numerical optimization of the log-likelihood. Specifically, for ten days of non-overlapping 30 minute windows, for each star centered at node v, we collect $\lambda_{vi} = \sum_{e \in outedges(v)} \lambda_{ei}$, where i is the sampling index, $(i = 1, \ldots, N)$.

The log-likelihood is given by

$$l(p_v, \tau_v, \eta_v | \lambda_v) = k \log p_v + (N-k) \log(1-p_v) + \sum_{i=1}^{N} \log f_\Gamma(\lambda_{vi} | \tau_v, \eta_v) I(\lambda_{vi} > 0) \quad (5.1)$$

where k is the number of positive λ_{vi} in the sample for star v and f_Γ is the Gamma density:

$$f_\Gamma(\lambda_{vi} | \tau_v, \eta_v) = \frac{\lambda_{vi}^{\tau_v - 1} e^{-\lambda_{vi}/\eta_v}}{\Gamma(\tau_v) \eta_v^{\tau_v}}$$

Let $(\hat{p}_v, \hat{\tau}_v, \hat{\eta}_v)$ be the mle's obtained from maximizing 5.1. Then for an observed λ_v, the upper p-value is given by

$$P(\Lambda_v > \lambda_v) = \hat{p}_v(1 - F_\Gamma(\lambda_v | \hat{\tau}_v, \hat{\eta}_v)$$

where $F_\Gamma = \int f_\Gamma$, the cdf.

5.2 Path p-values

The large number of paths makes modeling λ_γ for each path prohibitively expensive, both in computation time and memory requirements. Instead, we build a model for each individual edge, and then combine them during the path likelihood calculation.

5.2.1 Modeling the Edge GLRT Distribution

For each edge e, let Λ_e have the distribution of e's glrt scores. Again, we use a zero-inflated Gamma distribution. Now, however, it will be on a per-edge basis. Let

$$\Lambda_e = B_e X_e$$

where $B_e \sim \text{Bernoulli}(p_e)$, and $X_e \sim \text{Gamma}(\tau_e, \eta)$, with edge specific shape τ_e and shared scale η. That is, we have two free parameters for each edge, p_e and τ_e, and a common scale parameter for all edges, η.

5.2.2 Estimation

We now describe estimation of p_e, λ_e, and η. To do so, we simplify notation by letting $y_e = [y_{e1}, \ldots, y_{ek_e}]$ consist of only the positive samples in λ_e, where k_e is the number of positive samples on edge e. Let the total number of samples on each edge be given by N (the same for all edges). Then the log-likelihood for a single edge is given by

$$\begin{aligned} l_e(p_e, \tau_e, \eta) &= k_e \log p_e + (N - k_e)\log(1 - p_e) + \\ &\quad \tau_e \sum_{i=1}^{k_e} \log y_{ei} - \frac{1}{\eta}\sum_{i=1}^{k_e} y_i - k_e \tau_e \log \eta - k_e \log \Gamma(\tau_e) + C \\ &= g(p_e) + h(\tau_e, \eta) \end{aligned}$$

where C is a constant and g and h are noted for convenience. Maximizing $g(p_e)$ with respect to p_e yields

$$\hat{p}_e = \frac{k_e}{N}$$

Maximizing $h(\tau_e, \eta)$ with respect to τ_e provides an estimate for a given η, but maximizing the joint likelihood over the pool of edges requires maximizing

$$\sum_e h_e(\tau_e, \eta) \tag{5.2}$$

We proceed iteratively. Initially, we fix $\eta = 2$, the χ^2 value for this parameter. To maximize τ_e for a given η, we numerically maximize $h(\tau_e, \eta)$ with respect to τ_e, yielding $\hat{\tau}_e^{(t)}$, where t denotes the iteration index. This provides a set of $\hat{\tau}_e^{(t)}$ values, one for each edge. Now, given this set, we maximize the joint likelihood over all of the edges, equation 5.2, with respect to η. This has a closed form solution:

$$\begin{aligned}
l(\eta) &= \sum_e h_e(\tau_e, \eta) \\
\frac{\partial l}{\partial \eta} &= \sum_e \frac{\partial l_e}{\partial \eta} \\
&= \frac{1}{\eta^2} \sum_e \sum_i y_{ei} - \frac{1}{\eta} \sum_e k_e \tau_e
\end{aligned}$$

Solving for η yields the update

$$\hat{\eta}^{(t)} = \frac{\sum_e \sum_i y_{ei}}{\sum_e k_e \tau_e}$$

Since the Gamma distribution is a member of the exponential family, the likelihood is convex, so that this approach hill-climbs within each iteration, resulting in hill-climbing overall. We cease the iteration when the sum of the relative changes between all parameters is small, and we denote the resulting parameter estimates as $\hat{p}_e, \hat{\tau}_e, \hat{\eta}$.

5.2.3 Obtaining a Path p-value from the Edge Models

Once the edge models are fitted, we have all of the information we need to calculate path p-values. Let $\Lambda_p = \sum_{e \in path} B_e X_e$. The 3-path exceedance p-value is the mixture exceedance given by

$$\begin{aligned} P(\Lambda_p > \lambda_p) &= \sum_{b_1=0}^{1}\sum_{b_2=0}^{1}\sum_{b_3=0}^{1} P(B_1 = b_1)P(B_2 = b_2)P(B_3 = b_3)P(\Lambda_p > \lambda_p | b_1, b_2, b_3) \\ &= \sum_{b_1=0}^{1}\sum_{b_2=0}^{1}\sum_{b_3=0}^{1} \left(\prod_{i=1}^{3}(1-\hat{p}_i)^{1-b_i}\hat{p}_i^{b_i}\right)\left(1 - F_\Gamma\left(\lambda_p | \sum_{j=1}^{3} b_i\hat{\tau}_i, \hat{\eta}\right)\right) \end{aligned} \quad (5.3)$$

where we used the fact that the sum of Gamma random variables is again Gamma.

5.3 Pooling of Sparse λ_e for Edge Likelihoods

Recall in Section 4.4 that we have two types of edges. Each edge in Edge Type I will have its own parameters, fitted from a historic set of λ_e. But when fitting λ_e for edges in Edge Type II, we use a communal pool, and the fitted model for λ_e is then used for every edge in Edge Type II, as one would expect.

To be complete, we mention the necessity of one additional edge pool. It is possible, even with a very large sample size, to obtain very few non-zero samples of λ_e, even in Edge Type I. Therefore, the estimation of the τ_e may be poor. These edges vary by the type of test we conduct in Chapter 6, but comprise between 0.5% and 5% of the Edge Type I edges. We create a pool of these edges' λ_e values, and estimate a single set of parameters for this additional communal pool of data.

5.4 Threshold Determination

To obtain thresholds, we simulate ten days of minute counts for each edge. We then slide 30 minute windows, offset by 10 minutes, over the ten days, calculating the minimum p-value in each window, just as would be done in the full scanning procedure. To achieve a false discovery rate of 1 alarm per day, we might take the tenth smallest p-value in the resulting list of p-values. But since the windows overlap, we choose to be less conservative, by counting minimum p-values resulting from consecutive windows on the same path as a single p-value, and find the tenth smallest minimum p-value associated with non-consecutive windows. In this way, alarms over several overlapping windows only contribute one alarm to the threshold determination. This follows what an analyst would do, which is to treat alarms in consecutive windows as the same anomaly.

Chapter 6

Simulation Setup

In this chapter we describe a series of simulations, applying the graph scanning approach to a variety of models and test types. In Chapter 7, we present the results of these various simulation/testing combinations.

6.1 Overview

For a given anomaly type and model, the simulation involves the following general steps:

1. For each edge, estimate historic parameters for this edge from the full 30 days of real data (See Section 4.4)
2. Fit models for the distribution of the λ_γ scores collected on the 30 days of data (See Sections 5.1 and 5.2)
3. If we are using paths to scan, obtain a screening threshold (See Section 6.2)

4. Obtain a p-value threshold from 10 days of simulated, non-anomalous data (See Section 5.4)

5. Simulate 100 days of minute data on each edge according to the historic estimates, except for the set of anomalous edges, where the model parameters are adjusted to introduce an anomaly (See Section 6.4)

6. For each of the 100 days of scanning

   (a) Slide a window of length 30 minutes over the day, offsetting each consecutive window by 10 minutes

   (b) Within each window, sub-select the edges of the entire data set for which there was at least one non-zero count in the window. This creates a subgraph of the overall graph.

   (c) For this subgraph, enumerate all 3-paths, and calculate their p-values. Note that the time windows are the same for each edge in each path, and consist of the entire 30 minute window.

   (d) If any path in this window has a p-value below the threshold, record all such paths, and examine no further windows for this day.

The idea behind step 6(d) above is that once an anomaly is detected, the system would pass the results to an analyst. This analyst would possibly shut the machines involved down, and determine what, if any, true malicious activity was present, before allowing the machines back on the network. Therefore, these first detection graphs are the only graphs we analyze in the results section, since for any further windows in the day, the anomaly would not be present in the data after forensic analysis was performed.

6.2 Screening the Full Path Calculation

When testing large numbers of small shapes, such as paths, computation speed is highly important. Complex calculations on a per-shape basis have a large effect on processing time. Even the calculation of the value in Equation 5.3 on every path is prohibitive if we hope to maintain real-time capability. Since there is a lot of information contained in the per-edge glrt, λ_e, it is attractive to use a screening process based upon these values. The edge p-value is given by

$$\rho_e = P(\Lambda_e > \lambda_e) = p_e(1 - F_\Gamma(\lambda_e|\tau_e, \eta))$$

where, again, p_e accounts for the mass at zero, and $1 - F_\Gamma(\lambda_e|\tau_e, \eta)$ is the exceedance probability of the Gamma distribution for that edge's likelihood score.

The product $\prod_{e=1}^{3} \rho_e$ of the edge p-values in the path is a measure of its anomalousness, which we can use to screen paths for the full path p-value calculation in Equation 5.3. We use 30 minute non-overlapping windows for the full 30 days, which provides 14,398 samples of this screening value for each estimated edge. The first percentile of the values is used as a screening value. Any product p-value smaller than this value is passed to the full p-value calculation. While a larger percentile could be used, the full minimum p-values determined using thresholding (See Section 5.4) agreed between these two screening thresholds. The full threshold resulting from using the .1 percentile also agreed with the 10 and one percentile full threshold values, but did not come with a significant increase in computational speed, so we use the first percentile screening threshold to be conservative.

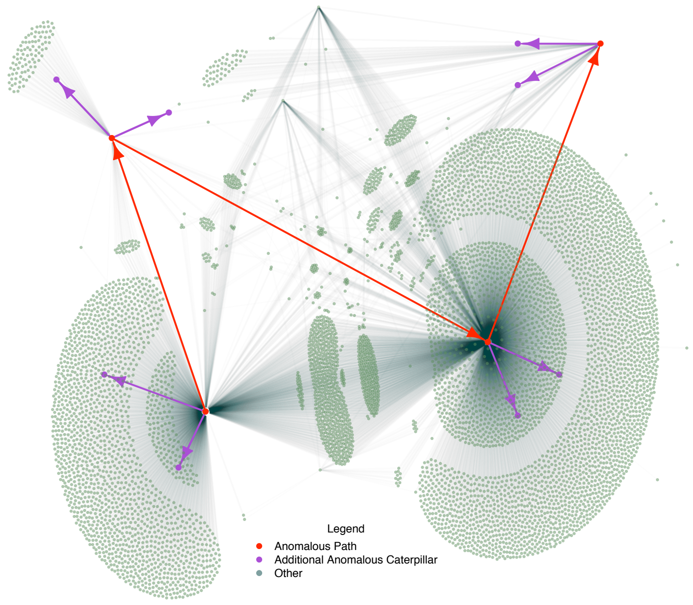
6.3 Simulated Anomalous Paths

For the simulation, we chose two paths for which anomalous counts would be generated, in order to better understand the system performance on anomalous paths which traverse different parts of the network. The first path, denoted path A, is meant to be representative of paths traversing through a set of heavily connected nodes. The second path, denoted path B, traverses more lightly used machines. To put these two anomalies in a visual context, see Figure 6.1. The red path in each figure indicates the true anomalous edges. The purple edges are part of a separate anomaly, the caterpillar, discussed in Section 6.5. We also provide the surrounding edges for context.

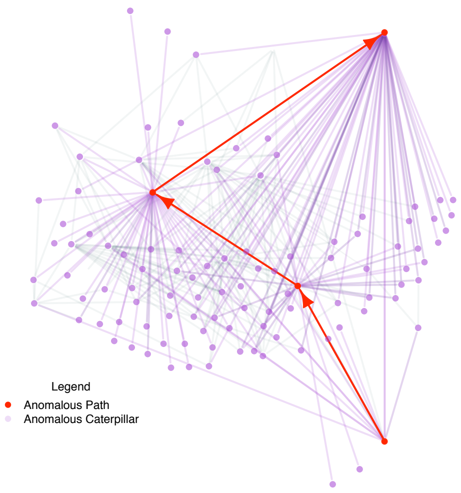
The number of in and out edges for each node (degree) in each of the two path types is summarized in Table 6.1. In addition, we have listed the Containing Graph Size (CGS). The containing graph is the union of all 3-paths in the graph which contain at least one of the truly anomalous edges. Thus, the CGS is the number of edges which are contained in any 3-path that passes through a truly anomalous edge. The average detected graph size (GS) in the simulation results, given in Tables 7.1 and 7.2 should be examined with this upper bound in mind. Observe that the CGS of Path A is roughly 39.4% of the entire edge set, while path B's CGS is only 1.6%.

	Node 1	Node 2	Node 3	Node 4	CGS
Path A	2640 / 240	141 / 89	5131 / 1446	95 / 65	220640
Path B	6 / 13	13 / 23	6 / 43	22 / 55	8931

Table 6.1: In / Out Degrees of Nodes in Anomalous Paths



(a) Path/Caterpillar A



(b) Path/Caterpillar B

Figure 6.1: Path and Caterpillar Anomalies. For each core path, the anomaly is plotted, along with the directly connected edges, for context. Red edges and nodes give the core path, and additional caterpillar anomalies are plotted in purple.

6.4 Anomalous Parameter Settings

We inserted anomalies in only one of these two paths at a time, by modifying only the estimated parameters in each edge of the anomalous path before simulating. Then, we simulate from each edge in the entire graph, using the modified parameters for the anomalous path, but the historic parameters for all other edges. For the OMM, we increased p_{01} by 0.2 from its historic rate. This increase was arrived at after consulting with cyber-security experts, whose intuition was that a likely hacker behavior could be to transition to the active state once every two minutes. We choose to be more conservative, insofar as making the anomaly even harder to detect, by inserting a one in five minute anomaly.

For the HMM, we introduce three types of anomalies. The first type of anomaly we denote as a P type anomaly, and consists of an increase in p_{01} of 0.2, for the same reason as the OMM. In addition, one could imagine a corresponding increase in the high-state mean. We therefore chose to introduce a second type of anomaly, denoted as an M type anomaly, consisting of an increase in the high-state mean by 1 count per minute. We reason that if the edge was active, the hacker's effect would be to increase the counts by 1 per minute. A final anomaly for the HMM model, denoted as a B type anomaly, consists of an increase to both p_{01} by 0.2 and the high-state mean by 1. This may be the most likely scenario in a real attack.

In order to understand the effects of these changes, we present the historic and anomalous parameters for each path type in Table 6.2. In Figure 6.2, we plot simulated data from the edges in Path A, inserting the various types of anomalies in different plots, to provide a visual display of the types of anomalies we are introducing on each edge.

Model	Path	Edge	p_{01} H	p_{01} A	p_{10} H	μ H	μ A	Size H
OMM	A	1	0.03	0.23	0.50	NA	NA	NA
		2	0.01	0.21	1.00	NA	NA	NA
		3	0.02	0.22	0.97	NA	NA	NA
	B	1	0.07	0.27	0.97	NA	NA	NA
		2	0.02	0.22	0.90	NA	NA	NA
		3	0.02	0.22	0.99	NA	NA	NA
HMM	A	1	0.01	0.21	1.00	3.44	4.44	432.84
		2	$3e$-4	0.20	$2e$-3	0.47	1.47	123.30
		3	0.02	0.22	0.97	3.43	4.43	19.07
	B	1	0.05	0.25	0.96	0.32	1.32	121.82
		2	0.04	0.24	0.67	0.50	1.50	150.84
		3	0.22	0.42	0.93	0.35	1.35	121.82

Table 6.2: Changes to Model Parameters for Anomalous Paths. H indicates historic parameters, A indicates anomalous parameters.

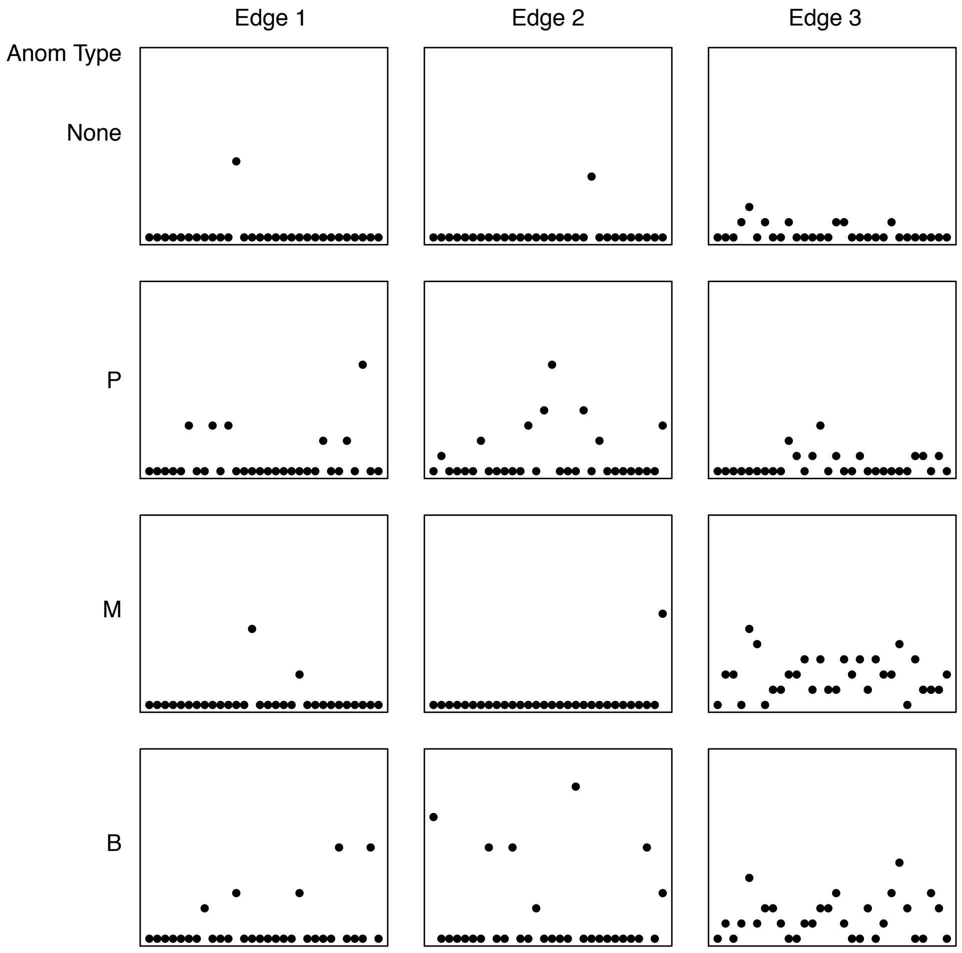


Figure 6.2: Historical and Anomalous Data Example. None: No anomaly. P: elevated p_{01}. M: elevated μ. B: both parameters elevated.

6.5 Other Shapes of Anomalies: Stars and Caterpillars

In Section 3.2, we discussed using star windows to scan. Here, we describe a star anomaly. Choosing one of the moderate out-degree nodes from Path B, namely Node 3 in Table 6.1, we introduce a Both Type anomaly on every out-edge from that node. Both path and star scanning is performed, and the results are described in Section

7.2.

In addition to a star anomaly, we introduced two types of caterpillar anomaly. Recall from Figure 3.1 that the caterpillar shape has a path at its core, and additional edges emanating from core nodes. The first caterpillar, denoted Cat A, has Path A as its core path, along with two additional anomalous edges emanating from each path node. Referring back to Caterpillar A in Figure 6.1, the red edges form the core of the caterpillar, but now we include the purple edges, to form this anomaly. This shape reflects a hacker behaving in a subtle way, by changing only a small number of edges around a path which is embedded in a much larger subgraph.

The second caterpillar, denoted Cat B, is also shown in Figure 6.1, and consists of Path B at its core, along with every out edge around this path's nodes, given in purple. This represents a hacker making many more anomalous edges per hop in the traversal, and the path is embedded in a much smaller subgraph than Cat A. These anomalies were designed to compare the performance of paths and stars on a mixed path-star anomaly.

The number of edges in each of these anomalies is given in Table 6.3.

Anom	Size
Star	43
Cat A	11
Cat B	174

Table 6.3: Graph Size of Star and Caterpillar Anomalies

6.6 Description of Tests

In addition to performing a variety of simulations, we also tested a variety of combinations of parameters. The test for each individual parameter was a restricted one. We look for increases in the parameters which could indicate hacker activity. Therefore, for testing p_{01}, we test $H_0 : p_{01} = \hat{p}_{01}$ versus $H_A : p_{01} > \hat{p}_{01}$. In order to calculate this restriction, we compute the unrestricted MLE, and if this value is less than $\hat{p}_{01}$, we set the estimate at the historic parameter, leading to a glrt value of 0. Testing for a change in the high state mean is entirely similar.

On the other hand, testing both parameters is more complicated. To test a change in both parameters, we first calculate the unrestricted MLEs in the window. If both are less than historic, then, as in the single parameter test cases, we set them equal to historic parameter values. If one of the parameters is less than historic, but the other is larger, we set the less than historic parameter at the historic setting, and re-maximize the likelihood of the other parameter. If both are larger than historic, we use these estimates. Thus, there is a subtle difference between the calculated score and the true GLRT, in the case when only one of the parameters was estimated at smaller than its historic value but the other parameter was then re-estimated. But the calculated values appear to remain sensitive to anomalies, as seen in the simulation results (Chapter 7) and in the real data example (Chapter 8).

For the OMM, we made p_{01} anomalous (on either Path A or Path B), and tested for this change. For the HMM, we can make a few more interesting combinations. In Table 6.4, we summarize the seven combinations that we used to analyze path scanning performance. To save space, we did not list Path A and Path B separately, but each of the tests in Table 6.4 was run using both Path A and Path B, separately.

Label	Parameters Changed		Parameters Tested	
	p_{01}	μ_1	p_{01}	μ_1
B-B	X	X	X	X
P-P	X		X	
M-M		X		X
B-P	X	X	X	
B-M	X	X		X
P-B	X		X	X
M-B		X	X	X

Table 6.4: Summary of Simulation and Test Combinations for the HMM. The Label column is a short-hand notation. C-T indicates Parameters C were changed and Parameters T were tested. B corresponds to a change in both parameters, P corresponds to a change in only p_{01}, and M corresponds to a change in only μ_1.

For both the star anomaly and the two caterpillars, only the HMM model was used. The anomaly we introduced changed both p_{01} and μ_1, and the test was over both of these parameters. However, for both caterpillar and star anomalies, we used both star and 3-path windows in the scan.

We emphasize that these simulations were performed using parameter estimates obtained from LANL's computer network, and consisted of over 250,000 individual models. Insofar as our models reflect the real data, these simulations provide a good feel for system performance in real settings.

Chapter 7

Simulation Results

In this chapter, we present the results of the scanning described in Chapter 6.

7.1 Path Scanning Simulation Results

First, we summarize the results of using path scanning on the simulated data. For both models, we present a box plot of the window of first detection, given in Figures 7.1 and 7.2. Recall that for each of 100 days, we ran the scanning on 30 minute windows, slid by 10 minutes, until we detected an anomaly. This first detection time is the statistic plotted in the box plots. The value to the right of each box plot, denoted PD, gives the percentage of the 100 days in which any window was detected.

Next, we describe the categories of detection summary statistics presented in Tables 7.1 and 7.2. Recall that when a detection occurs, it is not simply a single instance of the scan shape in the graph which resulted in the detection, but a union of the detected shapes, each of which had a p-value smaller than the false discovery threshold. It is this union we call the *detected graph* in Tables 7.1 and 7.2. Each value is an average over all 100 graphs, with standard errors in parentheses.

The first four categories, ANY, ALL, EXACT, and ONLY (described in Table 7.3), are the average of binary values summarizing the extent of the detection of the true anomalous path in the context of each detected graph. It is not always the case that any true anomalous edge was detected, but if so, then ANY is set to 1, and we can then ask whether or not ALL true edges were detected. If so, then we may ask if the EXACT path was a detected path. Finally, if this is the case, we can ask whether or not the true path was the ONLY path detected.

The next category, AEF, is the ratio of unique true anomalous edges to unique detected edges in the detected graph. GS is the average size of the detected graph. Since this graph is presented to an analyst for further forensics, these are both important measures of the system performance. A larger AEF value implies a more concentrated set of truly anomalous edges, and would lead to more efficient forensic examination by an analyst. Similarly, larger GS values indicate that a larger forensic effort is required.

MINP and MINF are binary measures of how well the true anomalous path stands out in the detected graph. MINP asks whether or not the path with minimum p-value is the true anomalous path. MINF asks whether or not the three most frequently detected edges comprise the truly anomalous path. While these are simple measures of how well the true anomaly stands out in the detected graph, in Section 7.2.3 we present a visualization of the detected graph, which we believe provides a more complete description of the detected graphs than these two summary values.

Finally, for the HMM box plots and summary table, we have columns on the left of each plot, which indicate the combination of type of anomaly introduced (Sim), parameters being tested (Test), and which Path was truly anomalous (Path). See Table 6.4 for definitions of the symbols B,P, and M.

7.1.1 OMM Tests

In Figure 7.1, we see the window of first detection box plot. In both Path A and Path B, a graph was detected in each of the 100 days. Detection occurred more quickly in the day for Path A than for Path B, and reflects the fact that there were many more paths traversing Path A than Path B, and therefore, we have more chances per time-window to detect Path A.

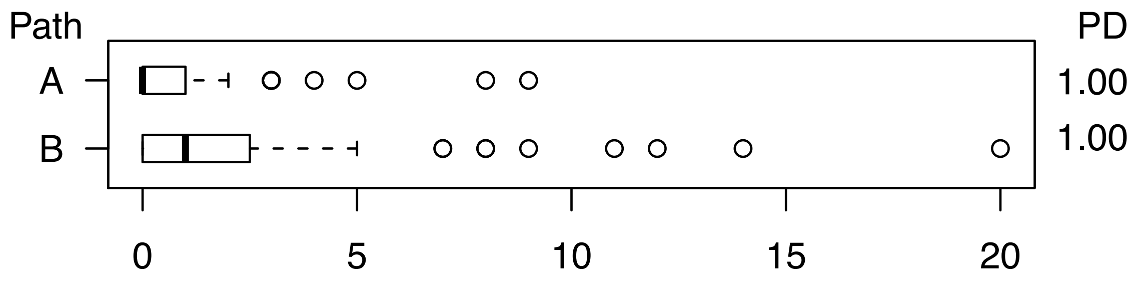


Figure 7.1: OMM Time to Detection. The x-axis represents the time window number of first detection. Since the windows are slid by ten minutes, these are in multiples of ten minutes.

Table 7.1 gives the summary of path scanning results for the OMM. As expected, Path B has fewer false paths detected, since it has fewer paths intersecting it. As a result, every category is improved. The model appears to reflect the simulated data fairly well, since detection rates are high. Nearly every graph contained at least one true edge, and, for Path B, over half of all detections had no false edges included (the

ONLY category). The GS was much larger for Path A, again, as expected. Still, referring to Table 6.1, even Path A's GS was far below the containing graph size (CGS). Path B detections were very tight in this respect, with a GS of 5.85. On average, that is, we only detected a graph about twice the size of the true anomalous graph (GS = 3). MINP and MINF are informative measures about the anomalous path. If the analyst conducting forensic examination of these results focused first on the lowest p-value path, for example, then he would be focusing on the true path nearly 90% of the time on Path B, and 62% of the time for Path A.

7.1.2 HMM Tests

We first examine the window of first detection box plots given in Figure 7.2. For Path A, anomalies were detected rapidly on every example except for the Mean anomaly. To see why this is so, refer to Table 6.2, and recall that in the Mean anomaly, p_{01} was not anomalous. We see that the historic p_{01} is very low on Path A. In addition, the historic μ_1 is relatively high on this path. These two parameters work in concert to make detection of an anomalous mean extremely difficult. First, since p_{01} is low, we do not have many opportunities to estimate the elevated μ_1, since we observe the high state very infrequently. In addition, since the historic μ_1 was already high, adding 1 to it to form the anomaly made for a relatively small change.

Nevertheless, Path A was consistently detected quicker than Path B on the same anomaly/test combination, and, again, this is explained by the fact that more paths intersect Path A, and therefore, we have more chances to observe Path A than Path B. Path B took far longer to detect, on average, than Path A on the p_{01} anomalies (Sim = P). This is in part due to the historically high p_{01} on Path B, as compared

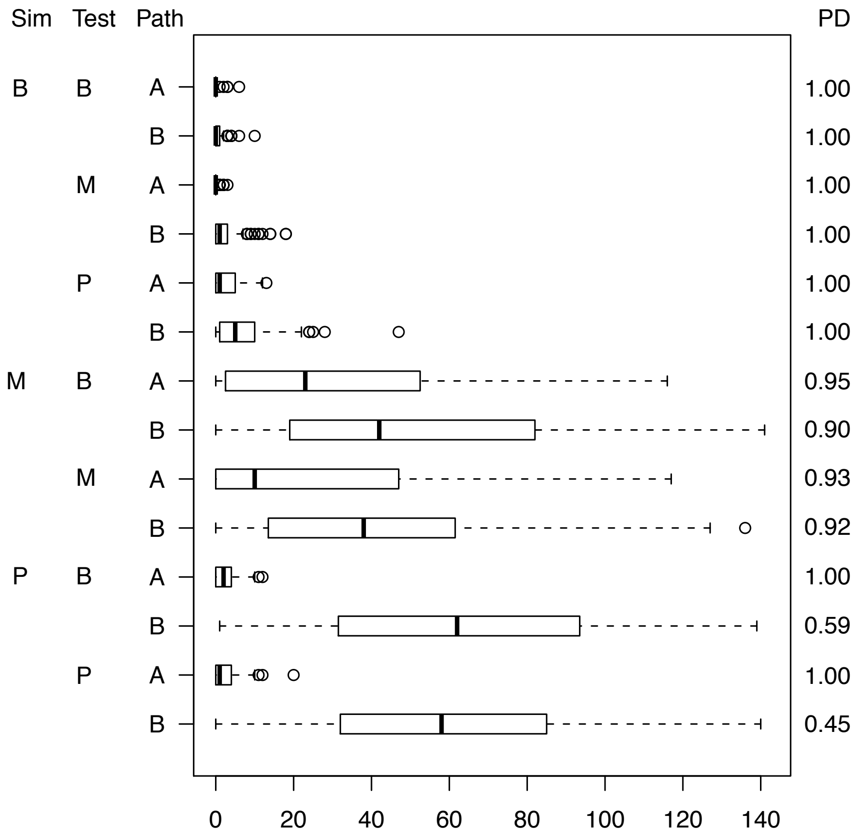


Figure 7.2: HMM Time to Detection. The x-axis represents the time window number of first detection, in multiples of ten minutes.

to Path A. Again, see Table 6.2. In addition, notice that edge 2 on Path B had a relatively low p_{10}. Therefore, if the edge enters the high state, it stays there for longer periods of time, and we have less chance to observe $0 - 1$ transitions in the window.

The PD values reflect the lack of detection for some tests, as well. Unlike the OMM box plots, we no longer detect a window every day. In general, though, this anomaly is much less severe than a corresponding change of 0.2 in the OMM, since this is a change to a hidden variable. Especially for the Sim P tests on Path B, whose PD values are down to 59% and 45%, we are not detecting very quickly at all. In fact, these reflect the false alarm rate, since no true detections were present on these tests (see the discussion below).

We now move to the summary of HMM detections given in Table 7.2. Overall, Path A had higher ANY scores than Path B, but more false edges were included in the detected graphs. Path B was detected exactly at a better rate, and Path B's detected graphs were more tightly focused on the true anomaly.

The B Sim tests were best detected, which is not surprising. Under this type of anomaly, testing both parameters is better than testing either individual parameter. The detected graph sizes are the largest under B Sim, since we are detecting the true edges better, and therefore we detect more paths that go through the true edges.

Next, we compare Sim B, Testing P with Sim P, Testing P. When both p_{01} and μ_1 are anomalous, we have more fidelity in testing p_{01}. This is because higher μ_1 values lead to more non-zero values when in the high state, making the estimation of p_{01} more accurate. Therefore, even when just testing p_{01}, we benefit from high μ_1 values. The same analysis applies to Sim B, Testing M versus Sim M, Testing M.

Now we look at differences between Path A and Path B on Sim P, Testing P or B. Path B did much worse. In fact, in both types of tests, no edges in Path B were detected. This is due to poor estimation of p_{01} on path B. Since Path B has relatively high historic p_{01}, and low historic means, we are not able to discern transitions as well as for edges in Path A, and estimating p_{01} is difficult. The OMM transition test performed better because when we see a 0, we know that the process is in the 0 state, whereas with the HMM, a 0 has a high probability under the high state parameters, on all three of path B's edges. This means that it is hard to tell the differences between 0s in the low state and 0s in the high state. Path A, on the other hand, has historically high means, and so the increase in transition rate stands out more.

Note that when anomalous data is simulated from a distribution with both parameters altered, we detect a p_{01} change quiet easily. This is due to the fact that estimation is poor in the HMM p_{01} when the high-state mean is close to 0, as we would expect.

Finally, MINP and MINF seem to have some discernment power for well detected paths, and these measures will be valuable in forensic analysis. Speed is of the essence during the forensic step, and any measure that helps the analyst focus is of value.

Path	ANY	ALL	EXACT	ONLY	AEF	GS	MINP	MINF
A	0.96(.02)	0.84(.04)	0.84(.04)	0.18(.04)	0.31(.04)	380.18(114.89)	0.62(.05)	0.57(.05)
B	0.93(.03)	0.93(.03)	0.93(.03)	0.56(.05)	0.73(.04)	5.85(.67)	0.89(.03)	0.86(.03)

Table 7.1: OMM Detected Graph Summary. Standard errors are given in parentheses.

Sim	Test	Path	ANY	ALL	EXACT	ONLY	AEF	GS	MINP	MINF
B	B	A	1.00 (.00)	0.98 (.01)	0.98 (.01)	0.06 (.02)	0.14 (.03)	970.38 (172.70)	0.79 (.04)	0.46 (.05)
		B	1.00 (.00)	1.00 (.00)	1.00 (.00)	0.21 (.04)	0.47 (.03)	10.89 (.73)	0.94 (.02)	0.92 (.03)
	M	A	1.00 (.00)	0.66 (.05)	0.66 (.05)	0.01 (.01)	0.06 (.02)	2143.44 (191.16)	0.01 (.01)	0.01 (.01)
		B	0.99 (.01)	0.97 (.02)	0.96 (.02)	0.39 (.05)	0.62 (.04)	7.75 (.64)	0.86 (.03)	0.93 (.03)
	P	A	0.99 (.01)	0.12 (.03)	0.12 (.03)	0.00 (.00)	0.22 (.03)	389.00 (117.95)	0.00 (.00)	0.04 (.02)
		B	0.97 (.02)	0.95 (.02)	0.95 (.02)	0.75 (.04)	0.87 (.03)	3.79 (.25)	0.92 (.03)	0.95 (.02)
M	B	A	0.76 (.04)	0.04 (.02)	0.04 (.02)	0.00 (.00)	0.06 (.01)	699.35 (129.06)	0.00 (.00)	0.00 (.00)
		B	0.57 (.05)	0.26 (.05)	0.26 (.05)	0.04 (.02)	0.25 (.03)	7.97 (1.52)	0.14 (.04)	0.22 (.04)
	M	A	0.72 (.05)	0.04 (.02)	0.03 (.02)	0.00 (.00)	0.05 (.01)	871.67 (154.21)	0.00 (.00)	0.00 (.00)
		B	0.51 (.05)	0.26 (.05)	0.26 (.05)	0.02 (.02)	0.18 (.03)	7.96 (.97)	0.13 (.04)	0.22 (.04)
P	B	A	0.99 (.01)	0.31 (.05)	0.31 (.05)	0.00 (.00)	0.17 (.02)	447.36 (134.97)	0.00 (.00)	0.08 (.03)
		B	0.00 (.00)	0.00 (.00)	0.00 (.00)	0.00 (.00)	0.00 (.00)	7.08 (2.26)	0.00 (.00)	0.00 (.00)
	P	A	0.98 (.01)	0.33 (.05)	0.33 (.05)	0.00 (.00)	0.14 (.02)	467.13 (134.04)	0.00 (.00)	0.10 (.03)
		B	0.00 (.00)	0.00 (.00)	0.00 (.00)	0.00 (.00)	0.00 (.00)	5.82 (1.36)	0.00 (.00)	0.00 (.00)

Table 7.2: HMM Detected Graph Summary. Standard errors are given in parentheses.

ANY	Percentage of time any true anomalous edge was present in the detected graphs	ONLY	Percentage of time only the exact anomalous path was present in the detected graphs	MINP	Percentage of time the true anomalous path had the minimum p-value among the detected paths
ALL	Percentage of time all true anomalous edges were present in the detected graphs (in any path)	AEF	Average number of unique true anomalous edges per Number of unique edges over all detected graphs	MINF	Percentage of time all of the true anomalous edges were detected more frequently than any of the other detected edges
EXACT	Percentage of time the exact anomalous path was present in the detected graphs	GS	Average size of the detected graphs		

Table 7.3: Key for Detected Graph Summary Tables

7.2 Star and Caterpillar Simulation Results

In this section, we discuss scan results on a star-shaped anomaly and two Caterpillar anomalies. In Table 7.4, we present a more terse summary of the detection statistics. Categories AEF and GS are as described in Section 7.1, but we chose to present the average percent of the full anomaly detected (PAD) in this summary. Since the star and caterpillar anomalies include many more edges, this statistic is more descriptive than in the path case, where the number of edges in the anomaly was always 3.

7.2.1 Star Anomaly

Referring to Table 7.4, it is clear that using star windows to scan a star anomaly is much more accurate than using paths. In fact, the star scan detected every true anomalous edge, and only those edges, for 99% of the days. In a single detection graph, the star scan detected a single additional edge. Keep in mind, however, that the star anomaly tested here was the full out-star of a given node. It is perhaps unlikely that a hacker would make every out-edge on a given node anomalous. If not, then the star scan will have less power, since it could include many non-anomalous edges. Regardless, paths picked up some portion of the anomalous star, at the cost of a much larger detected graph.

7.2.2 Caterpillar Anomaly

Next, we discuss the caterpillar results. Recall that Cat A is a very light anomaly (only 11 edges) whose core is a very well connected path. Path scanning detected the anomaly on the first window, but stars had a non-trivial time to first detection,

Anomaly Type	Scan Type	AEF	PAD	GS
Star	Path	0.18(.02)	0.23(.03)	448.50(106.49)
Star	Star	1.00(.00)	1.00(.00)	43.02(.02)
CatA	Path	0.01(.01)	0.79(.01)	3431.71(279.11)
CatA	Star	0.02(.00)	0.19(.01)	62.42(4.06)
CatB	Path	0.24(.01)	0.92(.01)	887.04(106.96)
CatB	Star	1.00(.00)	1.00(.00)	134.02(.02)

Table 7.4: Star and Caterpillar Detected Graph Summary

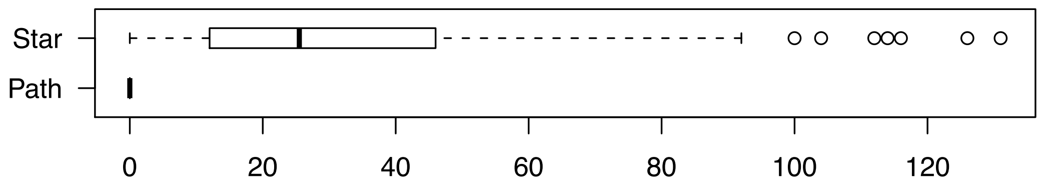


Figure 7.3: Caterpillar A Time to Detection. The x-axis is in ten minute intervals.

as seen in Figure 7.3. While the AEF value was fairly low for paths, on average, nearly the entire anomaly was detected.

The star scan, on the other hand, consistently detected only one of the stars in the caterpillar. This star was centered at Path A, Node 2 (as defined in Table 6.1), and was well detected because the second edge had extremely small historic p_{01}, as well as very small mean, as can been seen in Table 6.2, in the row corresponding to HMM Path A, Edge 2. The other two stars, and core path edges, were not detected at all by the star scan. In Figure 7.4, we provide a visualization of the star scan of the Cat A anomaly. One can see that most of the anomalous edges were missed, and many false edges were detected. This plot is to provide context for the star that

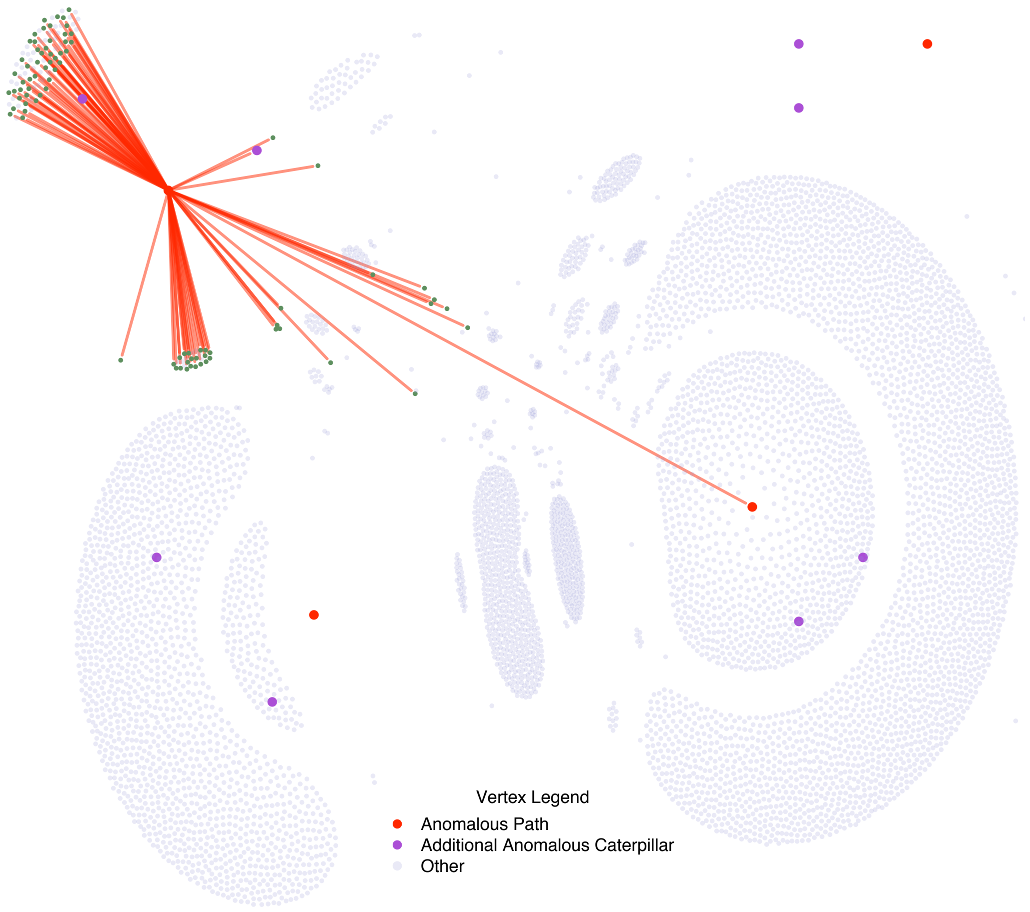


Figure 7.4: Star Scanning Results for Caterpillar A. Detected edges are plotted in red.

was detected, but for similar visualizations of the path scans on the caterpillars, see Section 7.2.3.

Recall that Cat B is a much heavier anomaly, involving every out edge of core Path B, for a total of 174 edges. But Path B is much more lightly connected in the graph, and therefore far fewer paths run through the anomaly than Path A. We might expect path scanning to suffer, as a result, since we test on fewer examples which encompass the true anomaly. However, path scanning performed even better than it did for Cat A, detecting more truly anomalous edges on average, and fewer falsely detected edges. Fewer false edges can be explained by the fact that fewer paths were inspected, but better detection of the true anomaly has to do with the difference between historic and anomalous parameters on the true anomaly. This is clear from looking at the historic versus anomalous parameter values, but since there were 174 sets of parameters to compare, we omit this analysis.

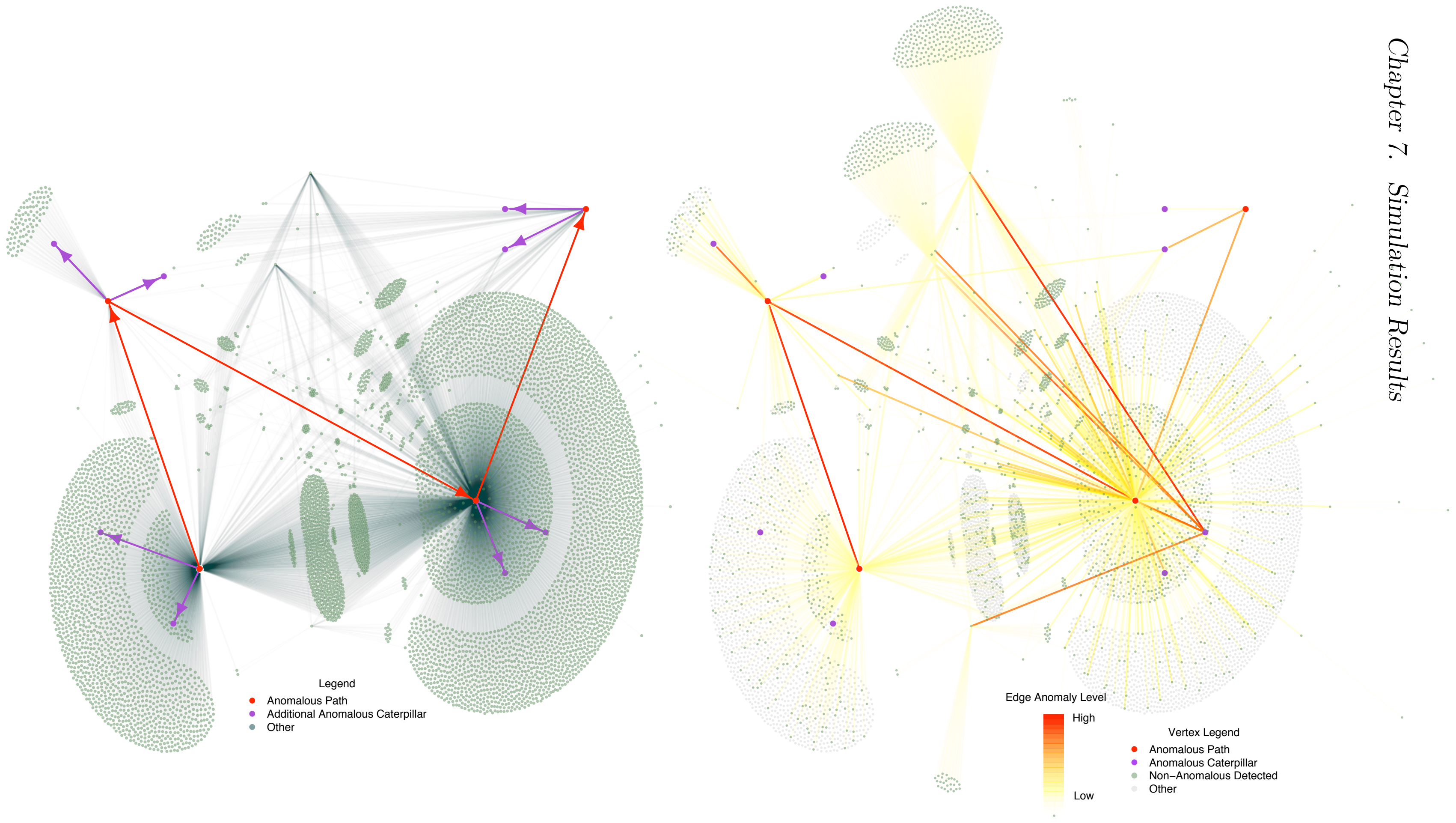
In the next section, we further analyze the detected caterpillar graphs, with a heatmap visualization.

7.2.3 Heatmaps on Caterpillar Detections

Recall that a detection in this setting corresponds to the union of every detected scan window. These unions may overlap on a set of edges. While stars do not overlap, paths do, and so, for each detected graph, we can count the number of times each edge appears in any detected path. This count can then be used to color edges in a heat map of the detection.

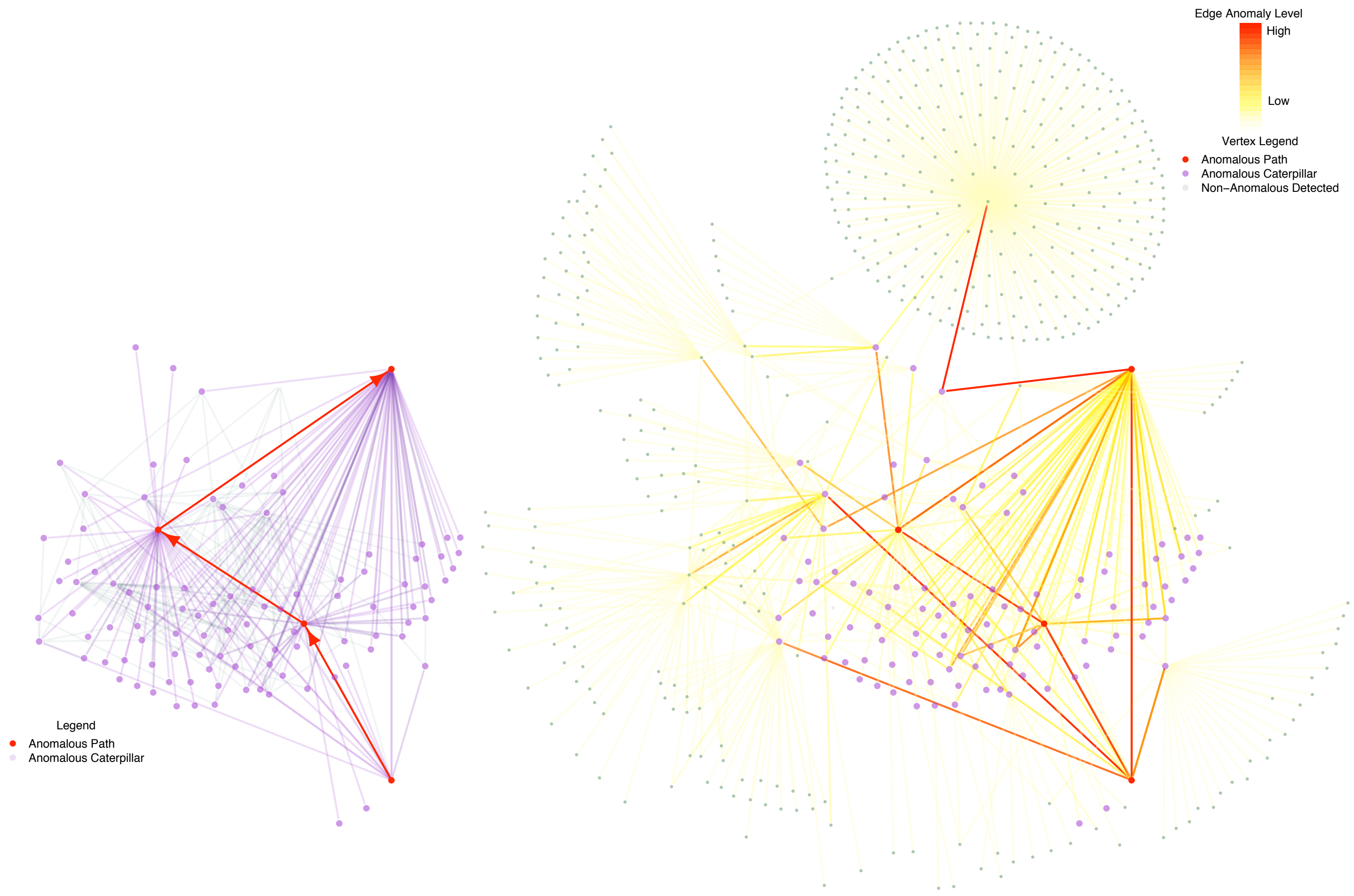
In Figure 7.5, on the left, we see Cat A embedded in its 1-hop containing graph. On the right, we see the path scan heat map of a single detected window. The core path is brightly colored, as these edges were detected very frequently. In addition, while they may be dim, for this detection at least, every true anomalous edge is present in this detection graph. These colors not only give the analyst an ordering of importance of the edges, but also provide an overall view of the structure of the anomaly. It additionally highlights the ability of paths to form more general shapes of detection than just the core shape.

In Figure 7.6, we provide a similar heat map of the Cat B anomaly. The core path was well detected, and most of the total anomaly was detected. There is a large star in the upper right of false detections. This is due to a highly connected node sharing an edge with the true anomaly, allowing paths through that node to intersect the true anomaly.



(a) Caterpillar A (b) Detection Heat Map

Figure 7.5: Anomaly Graph and Heat Map For Caterpillar A. The true anomaly is given on the left, with anomalous edges colored red and purple. The detected heat map is displayed on the right, with darker red indicating more evidence of an anomaly.



(a) Caterpillar B

(b) Detection Heat Map

Figure 7.6: Anomaly Graph and Heat Map For Caterpillar B. The true anomaly is given on the left, with anomalous edges colored red and purple. The detected heat map is displayed on the right, with darker red indicating more evidence of an anomaly.

Chapter 8

Real Data Detection

Since our goal with this work is a system which runs in real time, on real networks such as LANL's internal network, we considered it an important milestone to run, at least in prototype form, a path scan on real data from such a network. Therefore, in this chapter we describe a scanning of data contained in LANL's historic data archives.

8.1 Model Choice

We chose to use the HMM models whose parameters were estimated from real data, starting January 30, 2010, and ending 30 days later, as described in Section 4.4. We chose to test for an elevation in p_{01}. Initially, we attempted a test of both parameters, but we encountered several problems with testing the high-state mean. For example, we came across several edges where the historical high-state mean was estimated at a value of nearly 2, and in testing on the real data, we found counts as high as

500. These counts happened extremely rarely; for the most part, the counts were around 2, but since the 500 counts did not appear in the training data, the historical variance was estimated as small. This led to numerical problems in estimation, since the high-state negative binomial with mean 2 had numerically zero probability of producing a 500 count. In addition, testing for a p_{01} change had good performance in simulation, especially when the mean was also anomalously high.

8.2 Implementation Details

Since we used simulated data to set screening and p-value thresholds in the simulations, we require new thresholds when preparing to scan on real data. Therefore, the next 10 days of data, starting March 2, and ending March 12, were used to obtain these thresholds, using a discovery rate of one detection per day. Finally, the next 20 days were scanned using 3-paths.

We note that completely unestimated edges did arise in this data set. For this example, we used these edges in enumeration, allowing estimated edges to be "bridged" by these unestimated edges in the paths. But we did not use the data on these new edges to contribute to the path glrt score. In Chapter 9.1, we describe the approach we will be taking with these new edges in the future.

Thirty-eight unique detections occurred in these 20 days. We attribute the increased detections to a deterioration of the model fits, as time progressed. In practice, these models would be updated as described in Section 9.1.7.

8.3 Description of One Detection

While many of these detections look interesting, we choose to describe one in detail. A heat map of this detection, which occurred on March 22, 2010, is provided in Figure 8.1.

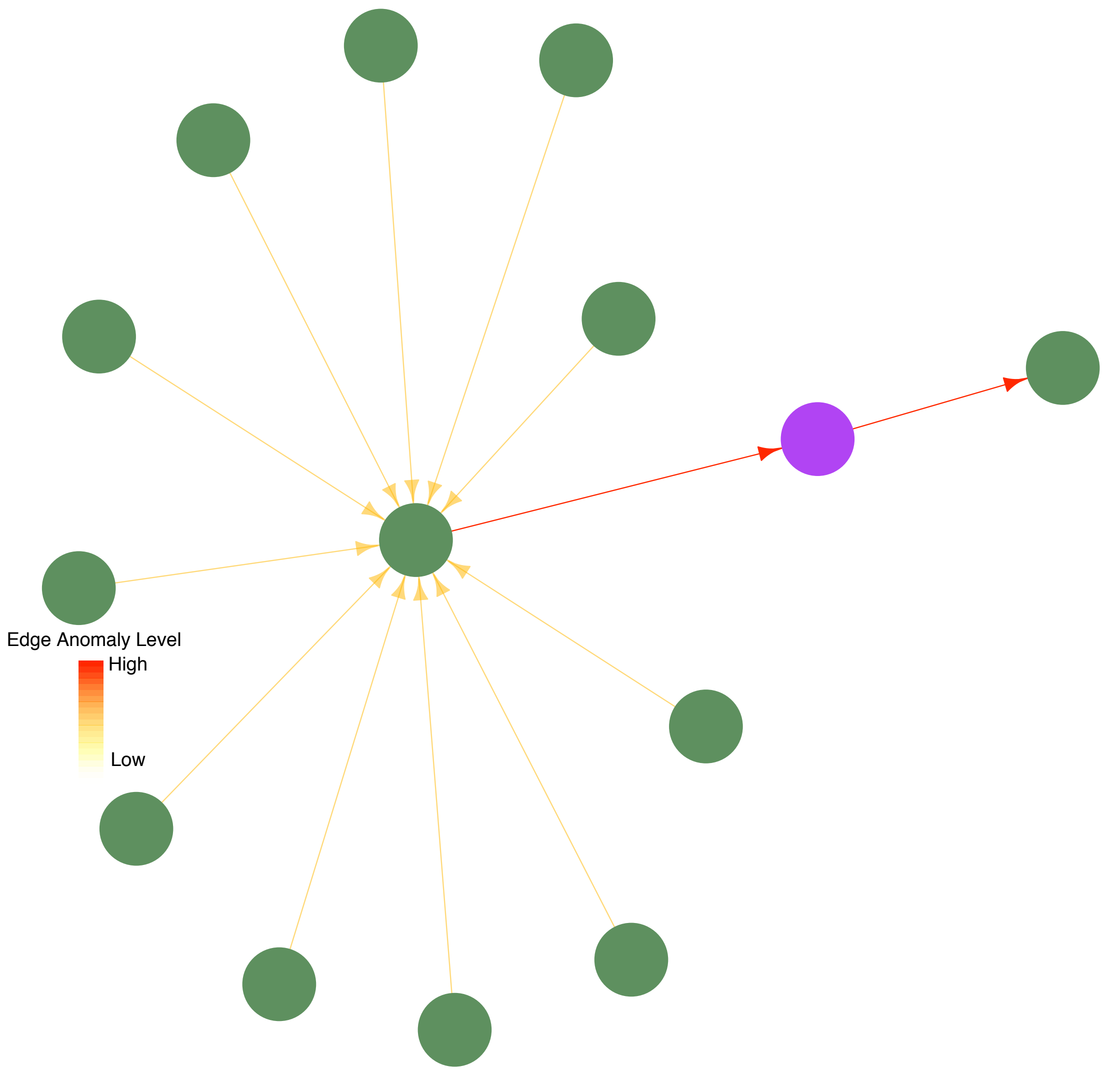


Figure 8.1: Real Detection Heat Map

In this figure, we see a star of 11 nodes around a central node, along with a 2-path (red) beginning at the central node. This central node is a meeting scheduling server, and the star nodes around it are user machines making connections to it to get the updated meeting schedule. The red edge leading out from the meeting server is an edge to a user machine, given in purple. The edge leading out from this user machine is an email server.

On March 22, at around 11:00 am, this graph was detected as anomalous. Each of the edges leading to the meeting server were identified once in the detected graph, and the two red edges were detected 11 times. This implies that each of the 11 3-paths starting at each star node all passed through both red edges.

When we conducted a forensic analysis of this graph, several facts emerged. First, the rate of counts on the two red edges increased significantly, while the edges leading into the star did not. This indicated an embedded anomalous 2-path in the 3-paths, which is apparent in Figure 8.1. Second, it was determined that the purple node changed significantly. Specifically, the purple node's IP address, which is used to label the node, did not change, but the actual computer registered to that IP changed. In addition, the user associated with that IP changed.

Since the user changed, the configuration settings when talking to the meeting server and email server on either side of the purple node changed. The new user began checking for updates from the meeting server much more often. In addition, the new user began checking email once every two minutes, instead of the historic setting of once in every 15 minutes.

While we were able to verify that this anomaly could be explained by normal network usage, it is nonetheless a very promising detection. Without the user change, this would be an extremely interesting anomaly, and one that our security team would investigate thoroughly. We have detected a local change in the graph, exactly as we set out to do. Since our goal was more than just an academic exercise, but an attempt to implement a real-world monitoring system that detects anomalies which are truly indicators of malicious behavior, we feel extremely encouraged by this result.

Chapter 9

Future Work

There is much that remains to be done in this work. Theoretical work has been done to improve the models and overall system performance, but since this work has yet to be realized in code, we present it as future work in this chapter.

9.1 Future Models for Edges in Computer Networks

In this section, we focus on the modeling of the communication patterns between two machines. In this dissertation, we have worked with observed and hidden Markov models to capture the switching behavior that is, in large part, the result of human interaction with the system.

Estimation and testing when hidden variables are involved, however, can be poor when we are interested in fairly short time-windows, since we have limited data to describe the full process. In particular, the active state may be rare, but, ironically, it is transitions to the active state, modeled with p_{01}, along with behavior in the active state, modeled with p_{10} in both the OMM and HMM, and μ_1 in the HMM, which we are most interested in testing. This is because the area of the distribution affected by a hacker acting in addition to the normal activity on that edge involves the active state.

In this section, we describe aspects of the data that need further attention, and outline a plan for future modeling.

9.1.1 Multivariate Data

So far, we have omitted the fact that there is, in fact, a multivariate data stream. For a given netflow record [26, 21, 3], we observe the following covariates:

- Client IP and Port
- Server IP and Port
- Start time
- End time
- Number of packets
- Number of bytes
- Layer 4 type (TCP,UDP,ICMP,etc.)

In this work, we used the server port to determine directionality. The server port is usually, but not always, the smaller port. Therefore edges originate at a higher port, and their destinations are the lower ports (exceptions to this rule are not discussed in this work, but are important, and we are developing a more complete heuristic for determining direction). We are not, however, using the fact that port information carries a likely protocol type. For example, SSH usually runs on Server Port 22.

There is a large set of literature on characterizing types of protocol traffic (for examples, see [32, 13, 22, 12]), but we do not choose to model each protocol individually. This is because a hacker may use any protocol in an attack, but he most likely will not use a protocol in exactly the way that that protocol is used by the user. It is the fact that the hacker is acting *in addition to* the normal activity of the user on that edge that we wish to capture. However, port information can be used to create pools of data to handle new edges (Section 9.1.2).

Currently, we use number of TCP connections closed per unit time (End time) as a response variable. Clearly, a more complete model would include duration and the numbers of packets and bytes. It is not yet clear whether Layer 4 type should be aggregated, or whether a categorical predictor variable should be incorporated into the model for this covariate.

9.1.2 New Edges

When a hacker gains a foothold in a network, he may not have perfect knowledge of the pattern of historic communications in that network. Therefore, he may make connections between hosts that have never communicated in the past. In fact, the very creation of new edges may be evidence of a lack of knowledge about how the network is normally used, which, in turn, could indicate an intruder's presence. It follows that the establishment of any new edges should pay a probability penalty in the anomalousness of the edge.

But new edges are somewhat tricky, since, by their nature, we do not have a historical model for them. One approach would be to pool data from previous new edges to estimate a model. But new edges do not necessarily have the same behavior as other new edges. For example, when a new server comes online, it will immediately start making high-rate activity on its out- and in-edges, whereas a new single-user machine would probably begin making new data on its edges at a much lower rate.

9.1.3 Daily and Weekly Patterns

In addition, the data exhibit daily and weekly patterns. As seen in Figure 4.1, the middle part of the work day has higher counts than at night. Different schedules on different days can also be seen. For example, the first Friday had activity, but the second Friday did not. This is due to an alternating Friday-off schedule that many employees have. Holidays are also an issue.

Unlike the models used in this work, the parameters should be allowed to change smoothly through the day, similar in spirit to the diurnal and weekly patterns modeled on telephone call networks presented in [16], and implemented in KATS (see Section 1.3.2). In addition, day-of-week effects should be taken into consideration.

9.1.4 Outline for Modeling the Data

For each time window, anomaly scores for a path are constructed by summing anomaly scores for each edge in the path. Here, we describe the construction of anomaly scores for edges.

For a given window of time, w, measures summarizing the behavior of an active edge will be formed, denoted by $s_w = (s_{1w}, \ldots, s_{mw})$. The summary scores can be simple ad-hoc summary statistics, or model based statistics, (e.g. the glrt scores, λ_γ, as discussed in Equation 2.1). For a given window, a list of these statistics could include

- Total number of flows
- Median/mean time between flows
- Mean packet count per flow
- Number of concurrent flows
- Mean duration of flows
- Number of transitions between zero counts and non-zero counts
- etc.

For a given summary statistic s_{jw}, let F_j denote its distribution in past (active) windows. An anomaly score for this summary might be based on its p-value, i.e. $p_{jw} = 1 - F_j(s_{jw})$. A combined summary score for this window across all summary statistics can be formed by combining the corresponding p-values. For example, $a_w = -2\sum_{j=1}^{m} \log p_{jw}$, which is Fisher's combination method [8]. Other combinations are possible. For instance, normal scores of the summary statistics may be used, i.e. $z_{jw} = \Phi^{-1}(F_j(s_{jw}))$. Letting z_w be the vector of normal scores, and $V(z_w) = \Sigma$, one could use $a_w = z_w^T \Sigma^{-1} z_w$.

The anomalousness of a_w, relative to its history, is based on the p-value of a_w. If our models for S_{jw} fit well, this distribution may have a known form. For example, using Fisher's combination method, under independence of the s_{jw}, a_w is distributed as a χ^2, random variable, and a Gamma could be used to provide some flexibility. For the normal scores, if the z_w are in fact Multivariate Normal, then, again, a_w is distributed as a χ^2 random variable.

For less amenable summary scores, we require a model for the historical distribution of a_w, denoted F_a, and the combined edge p-value is $1 - F_a(a_w)$. This is exactly what was done in this work, using a zero-inflated Gamma, as described in Chapter 5. The distribution of the summary scores and a_w are conditional on the edge being active. The combined edge anomaly can be augmented with the probability of the edge being active in a given window.

9.1.5 Implementation

We can accomplish the task of calculating p-values for s_{jw} and a_w by fitting some parametric distribution (e.g. a Gamma) to the summary statistics. This will need to be done for model-based summaries as well as simple summaries, due to model deficiencies. The reference distribution may depend on time-of-day and day-of-week effects, which can be included as covariates. In addition, auto-regressive coefficients will be used to allow the probability of the window being active to be higher if previous windows had elevated probability of activity. Different distributions may be appropriate, depending on the nature of the summary as well as its assumed values. For example, a negative binomial distribution for count summaries is perhaps more appropriate, or more general discrete-time hazard models.

Regardless, pooling across edges too sparse to support their own estimation, and across entirely new edges, will have to be done. For now, for modeling sparse edges and new edges, we will pool according to protocol, since protocol-specific traffic should be somewhat self-similar, and borrowing data from other edges should be reasonable.

Further down this line of reasoning, a random effects model could be used to build a parent distribution for the parameters, and each individual edge would get parameters drawn from this parent distribution. This parent distribution provides for easy updating, and new edges are trivial to handle, since their parameters will just have the parent distribution as their prior. Similar remarks apply to the estimation of F_a, but in concept, F_a should be easier to estimate since presumably, the day-of-week and time-of-day effects should be largely removed by the modeling of s_{jw}.

9.1.6 Reference Distribution Estimation

We describe a reference distribution for S_w (omitting j). Suppose, first, that S_w is a count random variable, (e.g. S_w measures total number of flows). We capture whether the edge in question has non-zero counts (active) or not (inactive) with a Bernoulli random variable Z_w which is 1 if the edge is active, and 0 otherwise. Let $p_w = P(Z_w = 1)$.

Next, for modeling positive counts when the edge is active we let $N_w - 1 \sim NB(\mu_w, s)$, a Negative Binomial with mean μ_w and size s. Since this random variable will handle only positive counts, we model the counts given that they are positive, by subtracting 1. For continuous summary statistics, a zero-inflated Gamma could instead be used.

Then a model for S_w is

$$S_w = Z_w N_w$$

We further model the parameters p_w and μ_w as

$$\text{logit } p_w = \mathbf{X}^T \beta \quad \log \mu_w = \mathbf{X}^t \gamma$$

where $\mathbf{X}^T = [\mathbf{X_d}^T, \mathbf{X_h}^T, \mathbf{Z}^T]$, $\beta^T = [\beta_\mathbf{d}^T, \beta_\mathbf{h}^T, \beta_\mathbf{z}^T]$, and $\gamma = [\gamma_\mathbf{d}^T, \gamma_\mathbf{h}^T, \gamma_\mathbf{z}^T]$.

Here, $\mathbf{X_d}$ and $\mathbf{X_h}$ could be vectors of indicator variables which capture day-of-week and time-of-day effects. For example, we could use $X_d^T = [X_{d1}, \ldots, X_{d7}]$ and $X_h^T = [X_{h1}, \ldots, X_{h24}]$ where X_{di} indicates day i, and X_{hj} indicates hour j. Another option could be smooth basis functions. In addition, p_w and μ_w may have dependence across windows, and Z captures this dependence. Specifically, $Z^T = [Z_{w-q-1}, \ldots, Z_{w-1}]$, to allow for auto-regressing over the previous q windows.

Let T_w be an observed window of data. By L, we denote the likelihood of p_w and μ_w (omitting w):

$$L(p, \mu | T) = (1-p)^{N-n_i} p^{n_i} \prod_{i=1}^{N} NB(t_i - 1 | \mu, s) I(t_i > 0)$$

where N is the window size, and n_i is the number of positive counts.

To estimate the parameters, we might maximize this function with respect to β and γ. But to enforce smoothness between time effects and previous windows, we also wish for the parameters to smoothly change between their respective times. For example, for day-of-week parameters, we would like the average changes between each day and its neighboring days to be small. That is, we would like to penalize

$$\beta_\mathbf{d}^T \mathbf{\Gamma_d} \beta_\mathbf{d} = \sum_{i=1}^{7} \sum_{j=i-1,i+1} (\beta_{di} - \beta_{dj})^2 \tag{9.1}$$

where $\mathbf{\Gamma_d}$ is defined below. In the above equation, arithmetic is done mod 7, so that $7 + 1 = 1$. This captures the cyclic nature of the days of the week. We enforce similar restrictions on ${\beta_\mathbf{h}}^T \mathbf{\Gamma_h} \beta_\mathbf{h}$. Finally, we would like the average size of the activity parameters $\sum_{i=1}^q \beta_{zi}^2$, to be small in order to regularize the auto-regressive dependence over past windows.

To enforce these conditions, we minimize

$$-\log L(p, \mu | T_w) + \beta^T \mathbf{\Gamma} \beta + \gamma^T \mathbf{\Phi} \gamma.$$

Here, $\mathbf{\Gamma}$ and $\mathbf{\Phi}$ provide the smoothing desired.

To be more clear, we further describe $\mathbf{\Gamma}$. Let

$$\mathbf{\Gamma} = \begin{bmatrix} \mathbf{\Gamma_d} & 0 & 0 \\ 0 & \mathbf{\Gamma_h} & 0 \\ 0 & 0 & \mathbf{\Gamma_z} \end{bmatrix} \quad \text{so that} \quad \beta^T \mathbf{\Gamma} \beta = \begin{bmatrix} {\beta_\mathbf{d}}^T \mathbf{\Gamma_d} \beta_\mathbf{d} & 0 & 0 \\ 0 & {\beta_\mathbf{h}}^T \mathbf{\Gamma_h} \beta_\mathbf{h} & 0 \\ 0 & 0 & {\beta_\mathbf{z}}^T \mathbf{\Gamma_z} \beta_\mathbf{z} \end{bmatrix}$$

We focus on the day-of-week effects, which involve ${\beta_\mathbf{d}}^T \mathbf{\Gamma_d} \beta_\mathbf{d}$. Letting

$$\mathbf{\Gamma_d} = \begin{bmatrix} 2 & -1 & 0 & 0 & 0 & 0 & -1 \\ -1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 2 & -1 \\ -1 & 0 & 0 & 0 & 0 & -1 & 2 \end{bmatrix}$$

leads to Equation 9.1, and $\mathbf{\Phi}$ is defined in a similar way.

9.1.7 Additional Future Modeling Features

While the above does provide a framework for capturing time-of-day and day-of-week effects, we desire a more complete modeling scheme. Pooling of new edge data to obtain estimates for a new-edge class, even if done by protocol, is somewhat heuristic. We intend to pursue a "parent" random effects model that will provide a distribution for parameters, from which we can draw to obtain new edge models. This parent model would make handling new edges and sparse edges fairly easy, since their parameters would use the parent model as a prior distribution. We would also like to add a variable that captures the probability of a new edge being created, since we believe an increase in new edge creation rate is an important aspect of anomalous activity.

Another issue is that of "stale" edges. These are edges which, for whatever reason, are no longer active. One or both machines on either side of the edge could have been taken down. Or the nature of those machines could have changed, so that the edge is no longer used. In any case, we wish to expire their parameters from the overall parameter set from which the parent distribution is formed. To do so, we will study heuristics for the expiration of edges.

Finally, we emphasize the "moving target" nature of the data. As the technology powering the network changes, and as the software running on machines evolves, we see a corresponding change in the data over time. A single model for a given edge for all time is not realistic. We require a method to quickly update parameters in models. To accomplish this, we will examine the use of weighted likelihoods to update parameters.

9.1.8 Aggregation of Bi-Directional Edges

Throughout this work, independence was assumed between edges. Most machines do not act as both a client and a server, so that it is rare to see a node with both edges leading to it and edges leading away from it. If two machines initiate connections to each other, however, the data on these edges may be highly correlated. To handle this, we will enumerate paths directionally as is done currently, but when a path contains a bi-directional edge, we will use a model of this bi-directional traffic, instead of treating each direction separately. This bi-directional model will then handle the correlations between the two directions, by aggregating their traffic into one variable.

9.1.9 Including Node Covariates in the Scan

It is the case that not only the communications (edges) between nodes are captured, but also the nodes themselves have behavior that can be measured. Examples include

- Average CPU load in the window
- Number of concurrent processes running
- Machine uptime (length of time machine has been powered on)
- Malware detection alarm state (do we believe the machine is currently infected with malicious software)
- etc.

Once we have a model for this node behavior, including the likelihood of node behavior for the nodes involved in paths in the scanning procedure is a straightforward extension of this work.

What is perhaps not straightforward is the collection of this data. Currently, data is collected at taps in the network, and not on individual machines. Our team, however, is currently working on a kernel level logging tool that could be installed on every host in the network, and that writes log events to a central storage point.

9.2 Ongoing System Improvements

In this section, we describe additional improvements to the system other than the modeling choices described above.

9.2.1 Improvements to Forensic Results

It is our hope that the heatmaps presented in 7.2.3 are useful in the forensic analysis of a detection. But the colors are not scaled to the number of paths going through each edge. That is, an edge could be red simply because it is very central (there are many anomalous paths running through it) in the detection. Scaling by the historical number of paths running through an edge can easily be done, and only requires calculating, for each edge, the number of paths in the historical graph that run through this edge.

In addition, when a set of paths is presented to an analyst, we provide a p-value ranking of the paths. However, this gives no indication of the absolute severity of a given detected path, but only relative to the other detected paths. For example, if we present an analyst with a path with a corresponding p-value of $1e - 20$, should the analyst immediately drop everything else and look at this anomaly? What about a p-value of $1e - 100$? In order to capture this idea, we may provide two classes of alarms. First, a "if the analyst have time" class, which consists of alarms below a threshold α_1, and a second class, "attend to this immediately", which has a lower threshold, α_2. These two thresholds could be set in the same way as our current threshold is set, that is, to produce alarms at a given rate. The threshold α_1 could be set at a one-per-day rate, and α_2 at a once per month rate, for example.

9.2.2 Scanning Multiple Shapes in the Cross Product Space

For the examples presented in this work, we used 30 minute time windows, and scanned using paths or stars. But attacks may be very short (minutes), very long (days), or somewhere in between. Longer windows, however, will have low power to detect short-duration attacks, and vice-versa. Therefore, we intend to run multiple scanners simultaneously, examining results from any individual scanner output. For example, we might use a 10-minute scanner, an hour scanner, a 6 hour scanner, and a day-long scanner, or some subset of these. In addition, to detect both path and star anomalies, we intend to run both types of scans, independently. The fast performance of the system should allow us to perform all of these scans with a minimal hardware investment.

Currently, time-windows are the same across all edges in a given graph window. Nested windows across a path, offset windows in time, or other combinations of time across the path should be investigated.

9.2.3 Experimentation to Improve System Performance

The system itself is in the prototype phase. We must run the system on a much larger set of data (e.g. 1-2 years worth), investigating the types of detections the system yields. In addition, we will work with analysts to improve models, reducing false positives if possible.

To accomplish this, we must, at the very least, have a method for parameter updating. While this is still in the research portion of the work, we mention an approach in Section 9.1.7. Once the parameter updating is implemented, we can run on these larger time scales without suffering from lack of fit as time progresses.

Since finding malicious activity is our primary goal with this work, we see running on large data sets as an important task, and it will be a major focus going forward.

Chapter 10

Conclusions

We have described a method for detecting anomalous activity where data is defined over time on edges in an underlying graph structure. We motivated the need for anomaly detection in this setting with the example of a hacker traversing a computer network. Attacks can be very localized, and so we introduce a method of windowing locally in the time $\times$ graph space. In each window, we calculate a scan statistic indicating whether or not the data in this local window is behaving according to a historic model.

We have introduced two types of graph windows for locality. Stars can be used to detect the event that a hacker has infected a central node, and is creating anomalies on every edge out of that node. A more subtle attacker might not produce anomalies on every edge, however. In addition, many attacks have, at their core, a path, which is the result of a hacker traversing through the network. Therefore, we have introduced k-paths as another type of graph window.

We have described an online system for acting on streaming data in real time.

The system is prototyped, and has been run on a variety of simulated and real data sets. The underlying algorithm for enumerating paths is extremely efficient, and we are pleased with the speed at which the system is able to enumerate many billions of local windows in the graph. We have discussed issues with calculating p-values of the likelihood scores, and discussed screening and thresholding of the p-values. This system will be instrumented on LANL's computer networks, and we are confident that it will increase the security of our communications.

We also discuss the results of simulations and a real-data example. The simulations provide insight into system performance on a variety of different anomalies and testing schemes. The real-data example is exciting, since we have detected the very activity we set out to detect. We presented heatmaps which should aid in the forensic investigation of detected graphs.

We have shown that paths can lead to very general shapes, and can detect, albeit not as well, star type anomalies. However, star shapes are excellent at identifying star anomalies. In mixed anomalies, such as caterpillars, we have shown that path scanning identifies the core of the caterpillar very well, while also identifying the 'fuzz' around the core path.

Additionally, we have described models that we believe will better reflect the data, and have attractive features, such as updating, modeling multivariate data streams, modeling day-of-week and time-of-day effects, and handling unestimated edges.

We are very pleased with the progress of this work, and are excited about the future. We believe we have built a fast, versatile anomaly detection system, which is based on real data and the input of cyber security experts. It is our intention to extend this work for the next few years, improving models, system performance, and forensic capabilities.

References

[1] Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society. Series B (Methodological)*, 57(1):289–300, 1995.

[2] Christopher M. Bishop. *Pattern Recognition and Machine Learning.* Springer, New York, NY, 2006.

[3] N. Brownlee, C. Mills, and G. Ruth. Traffic flow measurement: architecture (rfc 2722), 1999.

[4] G. Casella and R.L. Berger. Statistical inference. *Duxbury Press*, 2001.

[5] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, 41(3):15, 2009.

[6] M. Collins and M. Reiter. Hit-list worm detection and bot identification in large networks using protocol graphs. In *Recent Advances in Intrusion Detection*, pages 276–295. Springer, 2007.

[7] A. Doucet, N. De Freitas, and N. Gordon. *Sequential Monte Carlo methods in practice.* Springer Verlag, 2001.

[8] S.R.A. Fisher. *Statistical methods for research workers.* Number 5. Genesis Publishing Pvt Ltd, 1932.

[9] S. Forrest, S.A. Hofmeyr, A. Somayaji, T.A. Longstaff, et al. A sense of self for unix processes. In *IEEE Symposium on Security and Privacy*, pages 120–128. IEEE COMPUTER SOCIETY, 1996.

[10] Joseph Glaz, Joseph Naus, and Sylvan Wallenstein. *Scan Statistics.* Springer, 2001.

[11] N.A. Heard, D.J. Weston, K. Platanioti, and D.J. Hand. Bayesian anomaly detection methods for social networks. *Annals*, 4(2):645–662, 2010.

[12] M. Iliofotou, M. Faloutsos, and M. Mitzenmacher. Exploiting dynamicity in graph-based traffic analysis: Techniques and applications. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, pages 241–252. ACM, 2009.

[13] T. Karagiannis, K. Papagiannaki, and M. Faloutsos. Blinc: multilevel traffic classification in the dark. *ACM SIGCOMM Computer Communication Review*, 35(4):229–240, 2005.

[14] E.D. Kolaczyk. *Statistical Analysis of Network Data: Methods and Models*. Springer, 2009.

[15] Martin Kulldorff. A spatial scan statistic. *Communications in Statistics- Theory and Methods*, 26(6):1481–1496, 1997.

[16] D. Lambert and C. Liu. Adaptive Thresholds: Monitoring Streams of Network Counts Online. *Journal of the American Statistical Association*, 101(473):78–88, 2006.

[17] D. Lambert, J.C. Pinheiro, and D.X. Sun. Estimating Millions of Dynamic Timing Patterns in Real Time. *Journal of the American Statistical Association*, 96(453), 2001.

[18] T. Lane and C.E. Brodley. Temporal sequence learning and data reduction for anomaly detection. *ACM Transactions on Information and System Security (TISSEC)*, 2(3):295–331, 1999.

[19] C.R. Loader. Large-deviation approximations to the distribution of scan statistics. *Advances in Applied Probability*, 23(4):751–771, 1991.

[20] Q. Lu, F. Chen, and K. Hancock. On path anomaly detection in a large transportation network. *Computers, Environment and Urban Systems*, 2009.

[21] G. Lyons. Network Working Group P. Amsden Request for Comments: 2124 J. Amweg Category: Informational P. Calato S. Bensley, 1997.

[22] J. Ma, K. Levchenko, C. Kreibich, S. Savage, and G.M. Voelker. Unexpected means of protocol inference. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 313–326. ACM, 2006.

[23] B. Mukherjee, L.T. Heberlein, K.N. Levitt, et al. Network intrusion detection. *IEEE network*, 8(3):26–41, 1994.

[24] J.I. Naus. Approximations for distributions of scan statistics. *Journal of the American Statistical Association*, 77(377):177–183, 1982.

[25] C.C. Noble and D.J. Cook. Graph-based anomaly detection. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 631–636. ACM, 2003.

[26] P. Phaal, S. Panchen, and N. McKee. Inmon corporations sflow: a method for monitoring traffic in switched and routed networks (rfc 3176). Technical report, Technical report, Internet Engineering Task Force (IETF), 2001.

[27] W. Pohlmeier and V. Ulrich. An econometric model of the two-part decision-making process in the demand for health care. *The Journal of Human Resources*, 30(2):339–361, 1995.

[28] Carey E. Priebe, John M. Conroy, and David J. Marchette. Scan statistics on enron graphs. In *Workshop on Link Analysis, Counterterrorism and Security at the SIAM International Conference on Data Mining*, Newport Beach, CA, 2005.

[29] LR Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[30] Stanford SNAP Graph Library Data Sets. http://snap.stanford.edu/data/index.html.

[31] W.H. Woodall and M.M. Ncube. Multivariate cusum quality-control procedures. *Technometrics*, 27(3):285–292, 1985.

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

[33] D.Y. Yeung and Y. Ding. Host-based intrusion detection using dynamic and static behavioral models. *Pattern Recognition*, 36(1):229–243, 2003.