Software Engineering Institute

Identifying Anomalous Port-Specific Network Behavior

Rhiannon Weaver

May 2010

TECHNICAL REPORT
CMU/SEI-2010-TR-010
ESC-TR-2010-010

CERT® Program

http://www.cert.org

Carnegie Mellon

This report was prepared for the

SEI Administrative Agent
ESC/XPK
5 Eglin Street
Hanscom AFB, MA 01731-2100

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

This work is sponsored by the U.S. Department of Defense. The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

This work was created in the performance of Federal Government Contract Number FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.

For information about purchasing paper copies of SEI reports, please visit the publications section of our website (http://www.sei.cmu.edu/publications/).

Table of Contents

List of Figures

List of Tables

Executive Summary

Anomaly detection systems are designed to help analysts and operators uncover attacks that cause unusual deviations from well-known typical or baseline behavior. Large volumetric anomalies, such as distributed denial of service (DDoS) attacks or internet-wide vulnerability scans, are easy to see once the attack is fully underway, but they are sometimes preceded by more subtle yet serious deviations in activity. This activity is even more difficult to detect when it occurs on only a handful out of thousands of monitored assets. But the task of scaling techniques to large networks while controlling false positives is often unaddressed in proof-of-concept anomaly detection methods.

Unusual port-specific behavior is an example of a subtle predictive indicator for the widespread release of new vulnerability exploits. The CERT Network Situational Awareness (NetSA) team at Carnegie Mellon University's Software Engineering Institute (SEI) scaled and implemented a statistical proof-of-concept methodology to identify anomalous port-specific network behavior among the 65536 ports used in daily internet connections. The goal of the case study and recommendations we present is to help high-level analysts and researchers develop plans for implementing, adapting, and maintaining large-scale operational anomaly detection systems.

A port is an integer value between 0 and 65535 that represents a logical connection place for Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) communications. Sudden jumps in incoming traffic volume to a single port, or slowly increasing trends over time, can indicate new or renewed interest in a vulnerability associated with that port. Internet-wide outbreaks of malicious activity associated with newly released exploits can be preceded by port-specific scanning from multiple hosts. Detection of this activity may help forewarn network defenders of an impending attack.

Port-specific network behavior also arises from user activity that either occurs infrequently or migrates among unreserved ports (ports that typically range from 1025 to 65535) to evade firewall rules. This activity, such as instant messaging or peer-to-peer file sharing, may represent only a small change in volume relative to daily or weekly trends, but, if isolated, it can pinpoint internal clients that are bypassing established security policies.

Each port in an active network can be characterized by a discrete sequence of measurements over time, called a *time series.* To identify port-specific behavior, we apply statistical trending and outlier detection to these time series. We then calculate a *Z-score*, which represents the "oddity" of the port activity as the number of standard deviations an observed value lies away from its expected value. In addition to calculating Z-scores, we also display diagnostic measures that help to determine whether an anomaly is a sign of an internet attack.

In 2008, we analyzed a test set of three weeks of hourly data and evaluated 351 instances of anomalous behavior. Less than 1% of events we flagged as anomalous were classified as statistical false positives (arising from cyclical user behavior that should have been included in the model). However, the utility of the remaining flagged events for security purposes was limited. 41% of events were traced back to peer-to-peer "ghosting" activity, in which an idle internal IP address suddenly starts receiving SYN packets from a number of external addresses known to also

be communicating on standard peer-to-peer ports. This pattern can arise when an internal peer-to-peer user shares a temporary IP address that persists in external trackers after the user has gone offline. Ghosting is characterized by a many-to-one relationship: many external hosts attempting to connect to one host on one port. Normal but infrequent user behavior, such as secure web sessions, FTP, and mail sessions, were also flagged.

Based on our evaluation and analysis of our port anomaly detection algorithm, we recommend the following for volumetric anomaly detection:

1. **Choose variables in context**. Simple measurements may be useful for very specific trend analysis, for example, on a single-host level. But as the scope of the network monitor increases, and as metrics aggregate over many nodes, interpretable and actionable anomaly detection should become more targeted in scope, with appropriate metrics chosen to give a useful volumetric view that highlights a specific type of anomaly. Multivariate methods can incorporate information from several metrics simultaneously and may be more useful than the univariate methods often cited in the literature.
2. **Carefully model both trend and residuals.** Statistical models like those developed in this report describe both trend and residual variation once trend is removed. In anomaly detection, the residual variation is important to model accurately, especially for describing which extreme values are nonetheless typical (*tails* of the distribution) and which are truly anomalous. It is most important that the model for these residual extreme values is accurate across the population of assets that are being monitored, before universal thresholding can be used. Building models to monitor thousands of assets requires extensive historical reference data, diagnostics designed to validate the model at extreme values, and an exploratory pre-implementation phase focused on ranking goodness of fit across network assets that will be monitored.
3. **Correct for multiple hypothesis testing.** When faced with the possibility of thousands of statistical tests to determine alerts, a method that controls for multiple hypothesis testing, such as Benjamini and Hochberg's [**6**] method for controlling the False Discovery Rate (the FDR method), or the FDR method coupled with control-chart methods, should be used to ensure that any system conform to pre-determined type I error rates. The FDR method is especially appealing because the algorithm is simple and easy to implement on a large scale.
4. **Provide adaptive, interpretable methods for model diagnostics and evolution.** Anomaly detection systems based on statistical models should include methods for diagnosing model misfit and for updating model parameters. Inline methods (such as exponential moving averages) can be used, but even these methods may not be robust enough to adapt the model adequately over time. Simulations and application to historical data can be used to devise schedules for model maintenance. Analysts and operators that rely on any large-scale anomaly detection system should be trained on the interpretation of model diagnostics, and on the procedures that should be taken when diagnostics indicate model mis-specification.

Abstract

Increasing trends in traffic volume on specific ports may indicate new interest in a vulnerability associated with that port. This activity can be a precursor to internet-wide attacks. Port-specific behavior can also arise from stealthy applications that migrate to different ports in order to evade firewalls. But detecting this subtle activity among thousands of monitored ports requires careful statistical modeling as well as methods for controlling false positives. The analysis documented in this report is a large-scale application of statistical outlier detection for determining unusual port-specific network behavior. The method uses a robust correlation measure to cluster related ports and to control for the background baseline traffic trend. A scaled, median-corrected process, called a *Z-score*, is calculated for the hourly volume measurements for each port. The Z-score measures how unusual each port's behavior is in comparison with the rest of the ports in its cluster. The researchers discuss lessons learned from applying the method to the hourly count of incoming flow records for a carrier-class network over a period of three weeks.

1 Introduction

Anomaly detection systems are designed to help analysts and operators uncover attacks that cause unusual deviations from well-known typical or baseline behavior. Large volumetric anomalies, such as distributed denial of service (DDoS) attacks or internet-wide vulnerability scans, are easy to see once the attack is fully underway, but they are sometimes preceded by more subtle yet serious deviations in activity. This activity is even more difficult to detect when it occurs on only a handful out of thousands of monitored assets. But the task of scaling techniques to large networks while controlling false positives is often unaddressed in proof-of-concept anomaly detection methods.

Unusual port-specific behavior is an example of a subtle predictive indicator for the widespread release of new vulnerability exploits. The CERT NetSA team scaled and implemented a statistical proof-of-concept methodology to identify anomalous port-specific network behavior among the 65536 ports used in daily internet connections. The goal of the case study and recommendations we present is to help high-level analysts and researchers develop plans for implementing, adapting and maintaining large-scale operational anomaly detection systems.

A port is an integer value between 0 and 65535 that represents a logical connection place for Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) communications. In a two-way, client-server communication paradigm, a client opens a connection on a local port, and sends a message to the server on one of its ports. Application protocols often reserve a specific port on which the server should listen for requests associated with the application (for example, web servers listen on port 80). These ports are known as *service* ports. See Section 2 for more information on network communication.

Sudden jumps in incoming traffic volume to a single service port or slowly increasing trends over time can indicate new or renewed interest in a vulnerability associated with an application running on that port. In some cases, internet-wide outbreaks of malicious activity associated with newly released exploits are preceded by port-specific scanning from multiple hosts. Detection of this activity may help forewarn network defenders of an impending attack.

Port-specific network behavior also arises from user activity that either occurs infrequently or migrates among ports to evade firewall rules. This activity, such as instant messaging or peer-to-peer file sharing, may represent only a small change in volume relative to daily or weekly trends, but if isolated, it can pinpoint internal clients that are bypassing established security policies.

With 65536 active ports, the task of modeling individual port volumes over time, and alerting on a port-by-port basis, can be prohibitive. Many volume measurements exhibit characteristics such as non-constant variance over time, self-similarity, seasonality, and non-normality, which make traditional time series models such as those detailed in Brockwell and Davis [**1**], difficult to apply. McNutt and DeShon [**2**] showed that activity from gateway servers and vertical scans from external hosts leads to correlation in flow volumes over time across multiple ports. Figure 1 shows an example.

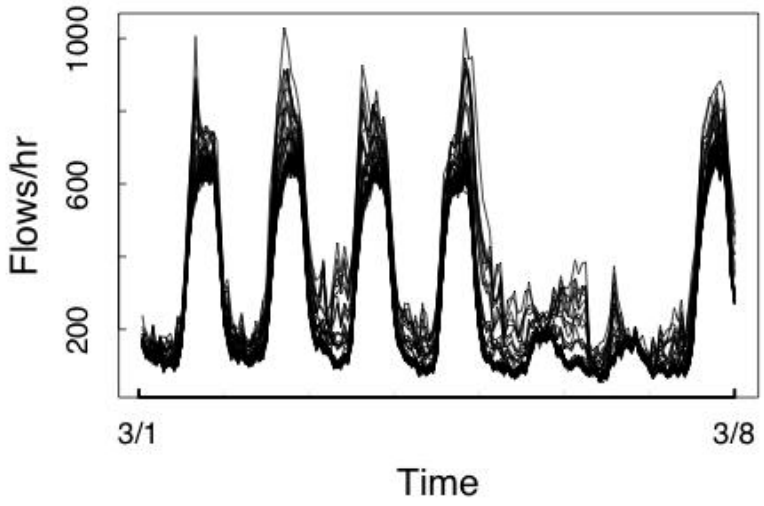

Figure 1: Flow Volume for Multiple Ports Exhibits Correlation over Time

The authors suggest clustering ports based on this correlation and using the median value per hour of each cluster as an expected value for the cluster's constituent ports (Figure 2). Using cluster medians to predict port volumes over time is advantageous because it can capture both smooth trends and global discontinuities across many ports, which would be difficult to capture with, for example, moving averages or autoregressive moving average (ARMA) models applied on a port-by-port basis. Furthermore, these methods do not rely on sequential storage of past data points for each series in order to calculate predictions.

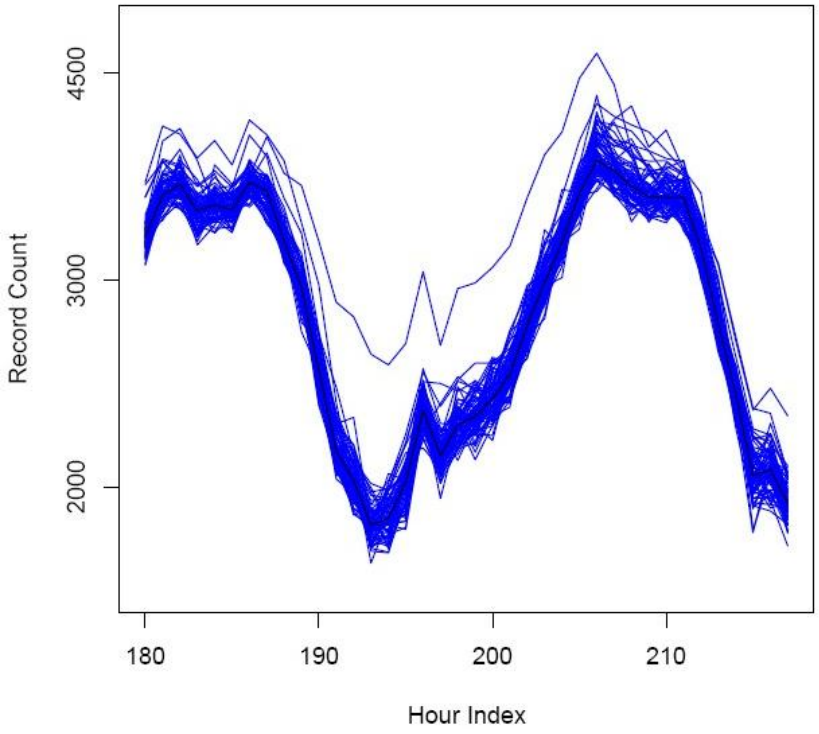

Figure 2: A Cluster of Flow Volumes of Related Ports and Cluster Median

A cluster of flow volumes of related ports (blue lines), and the associated cluster median (black line). An anomalous 16-hour surge occurs from hours 192 through 208.

Toward the goal of implementing this approach as a large-scale operational alerting method, this report defines the following formal methods for application to network flow data:

- **Robust, efficient clustering**. We employ a method called Minimum Covariance Determinant (MCD) correlation, from Rousseuw and Van Dreisen [**3**], to cluster time series of port volumes on the log scale in the presence of outliers. MCD correlation is used as a distance metric in single-linked agglomerative hierarchical clustering of groups of adjacent ports.
- **Statistical models for residuals**. Post-clustered residuals are standardized to follow a standard normal distribution (zero mean and unit variance), using a model that estimates a constant offset from the median for each port and variability related to individual port fluctuations—peak versus off-peak work hours, weekdays versus weekends, and volume of the cluster median. The resulting standardized residual is called a *Z-score.* Standardization of Z-scores across ports and clusters allows for application of a single threshold to all ports simultaneously to determine anomalous behavior.
- **Multiple hypothesis testing**. Alerts are generated via hypothesis tests—comparison of Z-scores to a threshold determined by probabilistic methods. When thousands of hypothesis tests are performed, the fraction of elements flagged due to random noise can result in many false positives. We suggest a tiered approach in order to reduce the number of hypothesis tests performed at any time period. Clusters are flagged based on an overall *cluster health* score determined via a chi-squared test. Once a cluster is flagged, the ports within are ranked by Z-score to determine the extent of anomalous behavior.

In Section 2, we introduce network monitoring via flow data and provide an overview of the relationship between port numbers, protocols, and client-server or peer-to-peer communication. In Section 3, we outline the formal port-specific anomaly detection method, with further details in the appendices. In Section 4, we present a case study of the method applied to three weeks worth of hourly incoming flow volumes to a large network. In Section 5, we discuss the implications of this case study on fully operationalizing the technique, as well as future directions of research.

2 Network Communication: Protocols, Ports, and Flows

The basic communication model for machines on the internet is a client-server connection. The client initiates a connection request in search of information, and the server responds to the request and provides the requested information. In the client-server model, each machine maintains its role, either client or server, for the duration of the communication. An alternative to this model is peer-to-peer connections, in which each machine is willing to act as either a client or a server, in order to share information or files across a large network of peers. Peers broadcast requests to the network, but they also listen for requests for information that they can provide.

The majority of client-server or peer-to-peer communications are negotiated using one of two protocols: TCP or UDP. Both TCP and UDP protocols require the assignment of a source port (from the initiator) and destination port (to the responder) within the connection. These ports are each represented as a 16-bit integer in the header for the communication protocol, hence the use of values from 0 through 65535. The port number is a logical representation of the connection; machines can use port numbers to keep track of multiple communications, either initiations or responses, at once. A machine or application that accepts communication requests to a specific port is said to be *listening* on that port.

Ports labeled from 0 through 1023 are known as *reserved* ports. Reserved ports are generally used only as destination ports in a client-server connection. Each reserved port is assigned a fixed application, common across networks and connections. For example, port 80 is reserved for HTTP connections, port 22 is reserved for SSH connections, and port 20 is reserved as an FTP control channel. Fixed or traditional application ports are not limited to reserved ports, however. A number of applications listen on ports greater than 1024; for example, port 6667 listens for IRC chat, port 5222 listens for the Instant Messaging protocol XMPP (run by Google Talk, among others), and port 8080 listens for HTTP proxies. In some cases, such services are registered with the Internet Assigned Numbers Authority (IANA), in which case these service ports in the 1024+ port range are referred to as *registered* ports.

While technically port 1024 could be used, the ports numbered 1025 through 5000 are used as source ports in client connection initiations by hosts running all versions of the Microsoft Windows operating system prior to Server 2008 and Vista. Windows Server 2008 and Windows Vista switched to using the range 49152-65535 (recommended by IANA for ephemeral port usage), as will subsequent versions of Windows. Specific UNIX-like operating systems use particular port ranges for initiating client requests (Macintosh OSX and BSD use 49152-65535; the Linux kernel 2.4 and 2.6 and Solaris 10 use 32768-65535). Client source port values are neither reserved for nor associated with any particular application, and these port values are meaningful only for the duration of the connection. Because of this, the client source port is often called an *ephemeral* port in a connection. Because ephemeral connection ports use the high-valued numbers 1024 through 65535, the port numbers themselves are also often referred to as *ephemeral ports*, even though many may also be associated with applications (registered or unregistered).

Port values play an important role in network flow monitoring. A *flow record* is an aggregation of volume, measured in both bytes and in packets, between two communicating network agents over time. Flows are defined and volumes are aggregated over short time intervals according to the IP addresses for both agents—the protocol used and, if the protocol is port-based, ports used for both agents as well. A unidirectional flow, used for example in the System for Internet-Level Knowledge (SiLK) flow collection system[1], counts the number of bytes and packets coming from a single source to a single destination. A two-way conversation is thus made up of two unidirectional flow records. Flow records can be used to count volumes and IP addresses associated with external connection requests to a single port in a monitored network. Although no payload data is recorded, the size, duration, IP addresses, and ports associated with the flow can give insight into the features of the communication and the applications used. A skilled analyst can use these features to draw inferences about the purpose of the communication and its underlying causes.

Scan activity appears as collections of flows showing low-volume (three packets or fewer) TCP connection requests, sent from a few external IP addresses to a wide cross-section of both active and inactive internal IP addresses, and receiving little to no communications in return. When scan activity increasingly targets a reserved or traditionally-used application port, it can indicate that attackers are interested in a newly discovered vulnerability. For example

- In November 2004, (CAN-2004-1080) a remote exploit for the WINS service was released, leading to a surge in activity on TCP port 42.
- In March 2005, (MS-ISAC #2005-004) a remote exploit for the Oracle FTP application was released, leading to a surge in activity on TCP port 2100.
- In June 2005, (CAN-2005-0773) an exploit for the Veritas Backup Exec Agent application, allowing for remote file access, led to a surge in activity on TCP port 10000.
- In May 2006 and January 2007, (SYM06-010 CVE-2006-2630) exploits for a buffer overflow vulnerability in the Symantec AV suite led to a surge in activity on TCP port 2697.

Of course, not all port surges are indications of malicious activity or exploits. Typical network behavior such as misconfigurations, large FTP file transfers, infrequent user behavior, and backscatter resulting from an IP address being used as a spoofed mail server can also lead to increased activity on a single port. Most of this activity can be characterized by the existence of either a single external host's or single internal host's involvement with the port-specific surge (many-to-one or one-to-many). One hallmark of vulnerability scanning is that, due to increased interest community-wide, it involves an increasing number of external hosts scanning a large number of internal hosts (many-to-many).

[1] Available at http://tools.netsa.cert.org/silk.

3 Modeling Approach

Adapting the proof of concept from McNutt and DeShon to a formal methodology for a port-specific alerting system requires four steps

1. calculating a correlation metric
2. clustering ports based on pairwise correlation
3. modeling port volume based on the median value of ports in a cluster
4. setting threshold values to determine anomalous ports

In this section, we address each of these steps in detail. In Section 3.1, we introduce notation for the methodological framework and discuss preliminary data transformations typical for count data. In Section 3.2, we introduce the Minimum Covariance Distance (MCD) method for robust correlation. In Section 3.3, we introduce local hierarchical clustering for finding groups of related ports. In Section 3.4, we discuss methods for modeling ports within a cluster. In Section 3.5, we discuss methods for determining alerts and measuring model health. In Section 3.6, we summarize the tuneable parameters, thresholds, and implementation details for the method.

3.1 Notation and Data Transformations

Let j represent a port assignment, ranging from 0 to 65535. For example, $j = 22$ refers to port 22, reserved for SSH communications using the TCP protocol. Let t represent a discrete time index, for example hours. Let X_{jt} refer to a univariate measure of volume that can be recorded for a port j at time t, for example, flow volume, or byte count. The values $X_{jt}, t = 1, 2, \ldots$ represent a stochastic process of volumes associated with port j over time. In notation, we will drop the time index t when referring to the process X_j as a whole.

Network data in the form of counts is often self-similar or *bursty* in arrival. This leads to time series X_j that exhibit skewed distributions with long tails spanning high values. Variability often increases as the count value increases, which makes interpretation of a correlation coefficient between two skewed time series difficult. Furthermore, these distributions tend to persist in residuals even after seasonal and daily trends are removed. The goal of analysis with Z-scores is to make use of thresholds based on Gaussian white noise with unit variance, as opposed to calculating thresholds based on empirical distributional percentiles, which require sorted data.

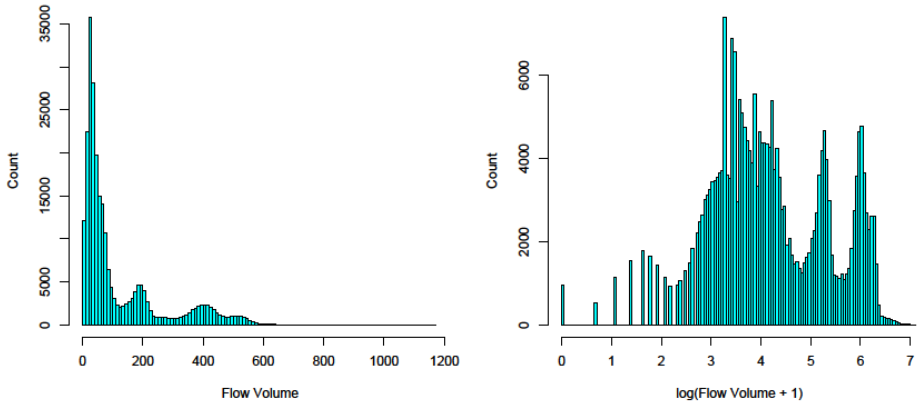

Figure 3: Histogram of Flow Volumes
Left: A histogram of flow volumes for a week of hourly counts on a random sample of ephemeral ports shows a long tailed, skewed distribution. Right: Log-transformed flows show a more balanced distribution over the range of potential values.

To ameliorate the effect of heavily skewed distributions, we change the scale over which the analysis is performed. This is done by transforming the counts to a less skewed distribution a priori, performing the analysis in the scaled space, and transforming conclusions back to the original scale. Nth-root or log transformations can be used in order to condense right-skewed distributions to appear more bell-shaped (see, for example, Chapter 12 in [**4**]). For port-specific anomaly detection, we use the transformed time series

$$Y_j = \ln(X_j + 1)$$

The values Y_{jt} account for skewness using the natural logarithm transformation. To prevent undefined counts for time indices t where $X_{jt} = 0$, each count is incremented by 1 before applying the logarithm. Figure 3 shows an example of the effect of the log transformation on hourly flow counts aggregated from a random sample of ephemeral ports.

In the course of this method, the set of ports j from 0 through 65535 are each assigned to one of the C clusters (which could also include clusters with only one element). Let c represent a cluster index ranging from 1 to C. Denote by $\{\boldsymbol{Y}\}^c$ the ports belonging to cluster c. Let Y_{kt}^c be the volume of the k-th port in the c-th cluster at time t. In notation, we drop the time index t when referring to the process Y_k^c as a whole.

3.2 Robust Correlation as a Distance Metric

In this section we develop the concept of robust correlation as a distance metric relating two port processes that can then be used as the basis for hierarchical clustering.

Suppose Y_i and Y_j $(i \neq j)$ represent the time series of log-volume metrics for ports i and j, respectively. Correlation measures how well the value of Y_i at any time t predicts the value of Y_j at time t (or vice versa), assuming a linear relationship. Correlation is a numerical value ranging from -1 (strongly negatively correlated) to 1 (strongly positively correlated). Values near 0

indicate a weak relationship. Figure 4 shows an example with $i = 46887$ and $j = 41292$ over one week's worth of hourly incoming flow counts.

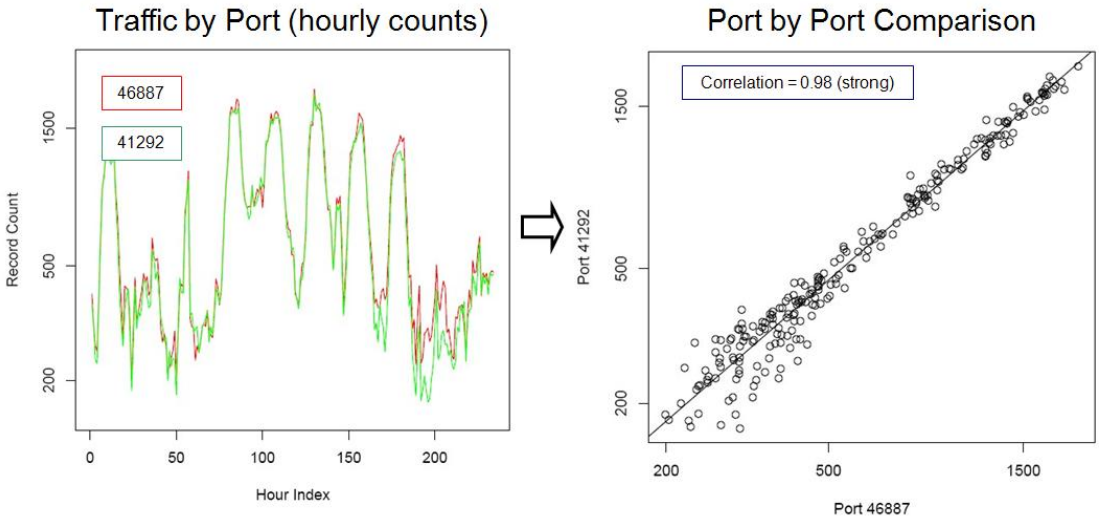

Figure 4: Two Ports Exhibiting Strongly Correlated Behavior (left) as Time Series and (right) in a Scatterplot

Mathematically, the correlation $\rho(Y_i, Y_j)$ between two processes is defined as

$$\rho(Y_i, Y_j) = \frac{Cov(Y_i, Y_j)}{\sqrt{Var(Y_i)Var(Y_j)}}$$

To estimate $\rho(Y_i, Y_j)$ from observed values at time points $t = 1, \ldots N$, we plug in the sample covariance and sample variance estimators to obtain

$$r_{ij} = \hat{\rho}(Y_i, Y_j) = \frac{s_{ij}}{\sqrt{s_{ii}^2 s_{jj}^2}} = \frac{\sum_{t=1}^{N}(Y_{it} - \bar{Y}_i)(Y_{jt} - \bar{Y}_j)}{\sqrt{\sum_{t=1}^{N}(Y_{it} - \bar{Y}_i)^2 \sum_{t=1}^{N}(Y_{jt} - \bar{Y}_j)^2}},$$

where $\bar{Y}_{\cdot} = \frac{1}{N}\sum_t Y_{\cdot t}$ is the sample average of values of $Y_{\cdot}$. Since port volume metrics are very rarely negatively correlated, we can use $|r_{ij}|$ as the basis of a distance metric for Y_i and Y_j

$$d_{ij} = d(Y_i, Y_j) = 1 - |r_{ij}|$$

The value d_{ij} is not strictly a distance metric (it fails the triangle inequality), but it can be used in clustering applications to describe a relative or predictive distance between two variables.

Using d_{ij} as a metric for clustering ports implicitly assumes that there is no anomalous behavior in any of the time points t that appear in the calculation of r_{ij}. In practice, it is often impractical to filter outliers or anomalies from the observed data by hand. In this case, a robust estimator of the correlation $\rho(Y_i, Y_j)$ is calculated using the following steps:

1. For an integer value $h < N$, find a subset of values $T_h \in \{1, \ldots, N\}$, such that
   a. T_h has h elements, and
   b. T_h minimizes the *trimmed covariance determinant*,
   c. $\delta_h(T) = s_{ii}^{T2} s_{jj}^{T2} - s_{ij}^{T2}$,

d. across all subsets of size h. Here, s^{T2} is the sum of squares obtained using only the h elements of the subset T, for example $s_{ii}^{T2} = \frac{1}{h-1}\sum_{k\in T}\left(Y_{ik} - \frac{1}{h}\sum_{k\in T} Y_{ik}\right)^2$.

2. Calculate the trimmed correlation

$$r_{ij}^h = \frac{s_{ij}^*}{\sqrt{s_{ii}^{*2} s_{jj}^{*2}}}$$

where $s^* = s^{T_h}$, and use the robust distance metric

$$d_{ij}^h = 1 - \left|r_{ij}^h\right|$$

This algorithm finds the subset of size h out of the N data points that maximizes the correlation between Y_i and Y_j. It is robust in that it excludes outlying points that may have arisen due to fluctuations or anomalies within the data set. A typical value for h would be, for example, $h = \lfloor 0.95N \rfloor$, yielding a 5% trimmed correlation.

As N grows, even a relatively large value of h can lead to a combinatorial explosion of sets over which to minimize $\delta_h(T)$. The Minimum Covariance Distance (MCD) method is a Monte Carlo algorithm that can be used to efficiently find a set T_h when the space of all possible subsets is too large to perform a systematic search. The MCD algorithm starts with a number ν of initial subsets, randomly chosen and refines these subsets in parallel using a method called the *C-step.* The refined subset with the lowest observed value of $\delta_h(T)$ is chosen as the subset for calculating r^h. Details of the algorithm are given in Appendix A.

3.3 Agglomerative Hierarchical Clustering

Agglomerative hierarchical clustering [**5**] (Chapter 12.3) builds a tree of relationships between series Y_i by successively merging individuals and clusters together that have the highest similarity. The algorithm uses the following steps to cluster M individuals:

1. Start with M clusters, each containing one individual, and an $M \times M$ symmetric distance matrix $\boldsymbol{D}$ whose ij-th element is the robust distance d_{ij}^h.
2. Find the nearest (most similar) pair of clusters. Say these clusters are $\{\boldsymbol{Y}\}^k$ and $\{\boldsymbol{Y}\}^l$. Denote by $d_{(k)(l)}$ the distance between $\{\boldsymbol{Y}\}^k$ and $\{\boldsymbol{Y}\}^l$.
3. Merge clusters $\{\boldsymbol{Y}\}^k$ and $\{\boldsymbol{Y}\}^l$ into a new cluster $\{\boldsymbol{Y}\}^{(kl)}$. Update the entries in the distance matrix $\boldsymbol{D}$ by deleting the rows and columns associated with $\{\boldsymbol{Y}\}^k$ and $\{\boldsymbol{Y}\}^l$, and adding a row and column denoting the distance of $\{\boldsymbol{Y}\}^{(kl)}$ to all other clusters.
4. Repeat steps 2 through 4 a total of $M - 1$ times, recording the identity of the clusters merged at each step, and the distances at which clusters are merged at each step.

This algorithm yields a tree of successive merges that consist of a single cluster of all individuals at the root, and M singleton clusters at the leaf nodes.

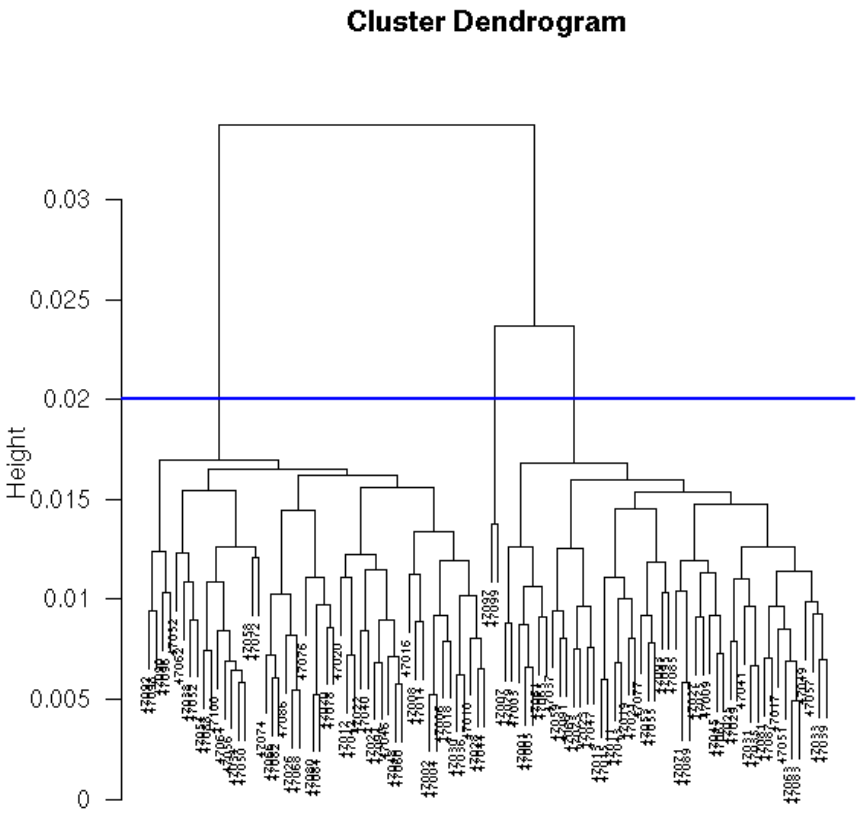

Figure 5: Dendogram Showing Hierarchical Agglomerative Clustering
A dendrogram showing the result of hierarchical agglomerative clustering performed on 100 adjacent ports. The cutoff point of 0.02 yields three clusters.

To determine distances between clusters with multiple elements, we use the complete linkage metric: For clusters $\{\boldsymbol{Y}\}^k$ and $\{\boldsymbol{Y}\}^l$, define $d_{(k)(l)}$ as

$$d_{(k)(l)} = \max_{Y_i \in \{\boldsymbol{Y}\}^k, Y_j \in \{\boldsymbol{Y}\}^l} d_{ij}^h$$

This distance ensures that at the point at which $\{\boldsymbol{Y}\}^k$ and $\{\boldsymbol{Y}\}^l$ are merged, all elements in the new cluster $C_{(kl)}$ are pairwise within the distance $d_{(k)(l)}$. This calculation requires pairwise comparison of all ports in each successive cluster, an operation requiring $O(n_k n_l)$ calculation time, where n_c is the number of elements in $\{\boldsymbol{Y}\}^c$. However, since d_{ij}^h does not satisfy the triangle inequality, bounds cannot be determined based on less compute-intensive processes, for example, comparison of cluster centroids, or of individuals to a cluster centroid.

Figure 5 shows a graphical representation, called a *dendrogram*, of the tree resulting from hierarchical agglomerative clustering of 100 adjacent ephemeral ports. Given a threshold distance d^*, the set of all merges that occur with $d_{(k)(l)} < d^*$ partitions individuals into clustered subsets.

To fully cluster all 65536 ports using pairwise distances would require over 2 billion calculations to populate a distance matrix, which is prohibitive in computational cost. Because ports tend to cluster with their neighbors, we use a heuristic, window-based approach to clustering these ports:

1. Define a maximum subset size M.
2. Starting at port 0, split the 65536 ports into successive subsets, with each set containing M adjacent ports.
3. Perform agglomerative hierarchical clustering within each of the $\lceil 65536/M \rceil$ subsets.

A typical value for M would be between 100 and 200. In practice, this method works well for ephemeral and *quiescent* ports that are not highly associated with a popular protocol. In public communication, peers suggested abandoning hierarchical clustering and simply using the adjacent subsets of size M as clusters. However, in practice, only a small number of ports (~ 25) are needed to define a cluster with stable median behavior, and clustering within subsets allows the threshold d^* to apply universally to clusters and removes potential outliers.

3.4 Modeling the Z-score

The purpose of clustering related ports is to remove time-dependent trends arising from scans, gateway activity, and other multi-port activity from the time series Y_j. For each cluster $\{\boldsymbol{Y}\}^c$, let $\tilde{Y}^c$ be the cluster median

$$\tilde{Y}_t^c = \text{median}_{Y_j \in \{\boldsymbol{Y}\}^c} \left(Y_{jt} \right)$$

For a port $Y_j^c \in \{\boldsymbol{Y}\}^c$, a natural choice for removing time-dependent trends is to study the residual process $\left(Y_j^c - \tilde{Y}^c \right)$. But this median-corrected residual process does not remove all sources of variation, nor does it standardize all time series to a standard normal process. Thus, for each port $Y_j^c \in \{\boldsymbol{Y}\}^c$, we define the median-corrected, scaled series

$$S_{jt}^c = \frac{\left(Y_{jt}^c - \tilde{Y}_t^c \right) - \mu_j^c}{\sigma_j^c}$$

The value μ_j^c is a baseline mean value estimated for each port using a trimmed average, excluding the top γ % of observations, where $0 \leq \gamma < 1$. This value accounts for a port's tendency to be either consistently higher or consistently lower than the cluster median. The value σ_j^c is a port-specific baseline standard deviation term, estimated from the data using a trimmed standard deviation based on the trimmed average. For a cluster $\{\boldsymbol{Y}\}^c$ with a relatively large number of ports, the median-corrected, scaled process follows a zero-mean Normal distribution across time values. Details for estimating μ_j^c and σ_j^c are given in Appendix B.

In practice, the baseline standard deviation σ_j^c does not completely capture the tendency for counts for a single port to have higher or lower variability depending on the time of day and on the overall port volume. For alerting, we use a scaled version of the median-corrected series called a Z-score

$$Z_{jt}^c = \left(\sigma_{jt}^s \, \sigma_{jt}^m \right)^{-1} S_{jt}^c$$

The values σ_{jt}^s and σ_{jt}^m are port-independent scale parameters based on cluster size, time of day, and cluster median. This correction allows us to compare the rescaled Z_j^c across many different clusters and ports, and to set universal threshold values when looking for anomalies. Details on

how the parameters σ_{jt}^{s} and σ_{jt}^{m} are estimated are given in Appendix B. We will denote by $\{\boldsymbol{Z}\}^{c}$ the Z-scores associated with cluster c.

3.5 Alerting

The basis of any anomaly detection system is an alerting method, but in a typical operational environment, this often requires thousands, if not more, of significance tests to be performed with each data update (for example by the minute, hour, or day). Alerting methods based on fixing the type I error rate (the proportion of false positives) in a hypothesis test must also control for this multiple testing environment or risk an explosion of false positives across all tests. For example, if 100 hypothesis tests are performed, each with a type I error rate $\alpha = 0.05$, then the expected number of false positives is equal to 5. And even a type I error rate of $\alpha = 0.001$ (1 in one thousand) across 65536 independent tests, for example using one test per port, results in approximately 65 false positives per cycle. If data is updated hourly, even this strict threshold can lead to a large number of false positives per day.

We address the multiple hypothesis testing problem in two ways. First, we use a cluster-based chi-squared test that allows for a tiered approach to alerting and reduces the number of initial tests necessary to discover anomalies. Second, we use the FDR method, as detailed by Benjamini and Hochberg [**6**], to determine the threshold at which to flag an alert.

For a cluster $\{\boldsymbol{Z}\}^{c}$ we define the chi-squared value at time t as

$$\chi_{ct}^{2} = \sum\nolimits_{Z_j \in \{\boldsymbol{Z}\}^{c}} Z_{jt}^{2}$$

This statistic can be used to test the cluster-based hypothesis

H_0: No unusual or anomalous activity is evident in the cluster, versus

H_a: At least one port in the cluster has anomalous traffic values (either high or low)

Under the null hypothesis of no unusual activity, χ_{ct}^{2} follows a χ^{2} distribution with degrees of freedom equal to the cluster size. For each cluster, we can use this distribution to calculate a p-value

$$p_{ct} = \Pr(\chi^{2} \geq {\chi_{ct}}^{2})$$

This hypothesis test rejects H_0 when the test statistic χ_{ct}^{2} is sufficiently large. This test is performed for each cluster, as opposed to each port, reducing the amount of simultaneous hypothesis tests that are performed. To control the average false positive rate among the multiple hypothesis tests at this stage, the FDR method is implemented as follows:

1. Choose a family error rate α.
2. For M independent hypothesis tests, sort the p-values from smallest to largest to obtain the ordered set $p_{(1)}, \ldots p_{(M)}$. Note that $a_{(j)}$ denotes the j-th order statistic of elements $a_1 \ldots a_n$.
3. Reject all m such that $p_{(m)} \leq \frac{m\alpha}{M}$.

Note that the test statistic χ_{ct}^{2} does not differentiate between a large contribution from one port and a series of smaller contributions from many ports. While the first situation is indicative of a

surge or volumetric anomaly, the second situation may be a sign that the model for the cluster (independent normal(0,1) residuals) is failing to account for trends or variation in the data. This failure is often referred to as *model mis-specification*. We use the term *cluster health* to describe the degree of model mis-specification evident in the cluster. A healthy cluster shows Z-scores following time-independent standard Normal distributions, with the possible exception of anomalies flagged due to surges in a few outlying ports.

When a cluster is flagged, we can then examine the individual Z-scores within the cluster to determine the source of the anomaly. For a cluster of J ports, we define the *nth-outlier* statistic as the value $Z_{(J-n)}$, the largest order statistic after the first n outliers have been removed. To search for high outliers corresponding to surging ports in a flagged cluster, we can compute a p-value for each ordered observation, based on the premise that $Z_{(j)}$ is the largest of j ports, using the formula

$$p_j = 1 - \left[\Phi\left(Z_{(j)}\right)\right]^j$$

where $\Phi(z)$ is the cumulative distribution function (CDF) of a standard normal (mean zero and unit variance) random variable. This p-value is conservative as compared with a joint p-value of all order statistics. However, when the highest order statistics are extreme outliers, and when the outliers represent only a small fraction of the cluster, the effect is negligible, and the p-value can be useful for ranking and flagging outliers. Ideally, port-specific anomalies would be a result of large contributions from only a few ports in the cluster. Large numbers of flagged ports (that is, over 30% of the cluster), or, alternatively, failure to detect any outliers in a flagged cluster, are signs of model mis-specification and poor cluster health. We present some examples in Section 4.3.2.

3.6 Summary of Tuning Parameters

Table 1 shows a list of the parameters that need to be set or tuned when implementing the port correlation algorithm, along with a set of suggested default values. The subset size M and number of MCD repetitions ν contribute to the computational burden of initially clustering ports, but they do not affect the speed of calculating Z-scores and flagging alerts, beyond the notion that M determines the maximum possible subset size. Depending on the initial noise of the data, the value of ν may be set much smaller than the recommended 500 repetitions by Rousseuw and van Driesen. As each pairwise calculation uses ν steps to calculate the MCD correlation, reducing ν can improve speed dramatically.

Table 1: Tuning Parameters for the Port Correlation Algorithm

Tuning parameters for the port correlation algorithm, with suggested default settings. Here, N is the number of data points recorded in the training set for each port.

Tuning Parameter	Symbol	Default
Subset size	M	150
Number of MCD repetitions	ν	500
Robust correlation threshold	h	0.95N
Clustering threshold	d^*	0.05
Port model trim percentage	γ	0.99
Alerting family error rate	α	0.05

The robust correlation threshold h and clustering threshold d^* can both be tuned to affect the predictive power in port clusters. A low value of h assumes that the training data is already noisy with events that cause surges in traffic that would correspond to a desired alert. High values of h assume that most of the training set contains only baseline, typical behavior. Lower values of d^* result in looser clusters of ports, consigning more variability to the model of typical behavior. This results in fewer alerts for low-volume surges. The family error rate α is the expected value of false positives as a percentage of the number of tests performed at each cycle. Low values of α also contribute to fewer flagged alerts.

Parameters that are estimated from the data, once tuning parameters have been set, include baseline means μ_j^c and standard deviations σ_j^c for each clustered port, as well as parametric models for estimating the scale parameters σ_{jt}^s and σ_{jt}^m. More detail on estimation is given in Appendix A.

4 Application to Flow Data

4.1 Data Collection

To test the methodology and alerting system, three weeks of data were collected from a large network with visibility across more than 24000 class C net blocks, each consisting of 256 IP addresses. The unit of measurement collected was a flow record, as described in Section 2, created using the SiLK flow collection system. To measure external interest in ports, flow records were collected that originated from external hosts on the internet directed into the network and aggregated by destination port. Counts of this incoming flow data were recorded for each of the 65536 ports on an hourly basis, for the period of April 1 through April 21, 2008.

To reduce the effects of backscatter and unsolicited replies, traffic collection included only incoming flows using the TCP protocol that did *not* have the following qualities:

- SYN ACK flags only set
- RST flag only set
- RST ACK flags only set

The week of April 1 through 7 was reserved as a training set, used to cluster ports and to estimate model parameters for each port cluster. The remaining two weeks were used as a test set, to flag alerts based on the models developed using the training data.

4.2 Parameter Tuning and Estimation

Port clusters were determined using a correlation distance threshold of 0.05 and an MCD threshold of 0.95 on 168 hours comprising the training data. Clustering was performed using the open source C Clustering library[2] (version 1.36) on a Linux cluster of 4 Intel 3GHz processors and 12GB RAM capacity. To determine the subset size M and MCD repetitions ν, taking into account operational computing time and robustness for the clustering metric, we performed benchmark tests on several combinations of parameters and compared clustering results for representative subsets from both the reserved and ephemeral range. Table 2 shows the results. The results suggest a feasible run time for subsets of 150 adjacent ports, with 30 MCD repetitions. This increases the number of pairwise comparisons calculated, with a number of repetitions chosen to increase speed while maintaining robustness of the MCD algorithm.

[2] Available at http://bonsai.ims.u-tokyo.ac.jp/~mdehoon/software/cluster/software.htm.

Table 2: Benchmark Results for Combinations of Subset Size and MCD Repetitions

Benchmark results for several combinations of subset size (M) and MCD repetitions (v). Parameters from the last row (*) were chosen for the training data, to keep the subset size large and cluster as many adjacent ports as possible, while conserving time.

M	v	Reserved Subset	Ephemeral Subset	Cluster Time per Subset	#Reserved Clusters	#Ephemeral Clusters	Estimated Full Clustering Time
100	100	0-99	45000-45099	3m 10s	77	4	34 hr
100	50	0-99	45000-45099	1m 45s	77	4	18 hr
100	30	0-99	45000-45099	1m 5s	77	4	11 hr
150	50	0-149	45000-45149	3m 55s	108	5	27 hr
***150**	30	0-149	45000-45149	2m 30s	107	5	18 hr

4.3 Summary of Results

4.3.1 Clusters

Using the correlation threshold of 0.05, 63476 out of 65536 possible ports were clustered into groups of two or more ports, yielding a total of 1127 clusters. Figure 6 shows a histogram of cluster sizes, with a minimum and mode of 2 ports per cluster, and a maximum of 150 ports due to the subset size M. Cluster sizes tend to favor three ranges: below 30, between 60 and 80, and above 125.

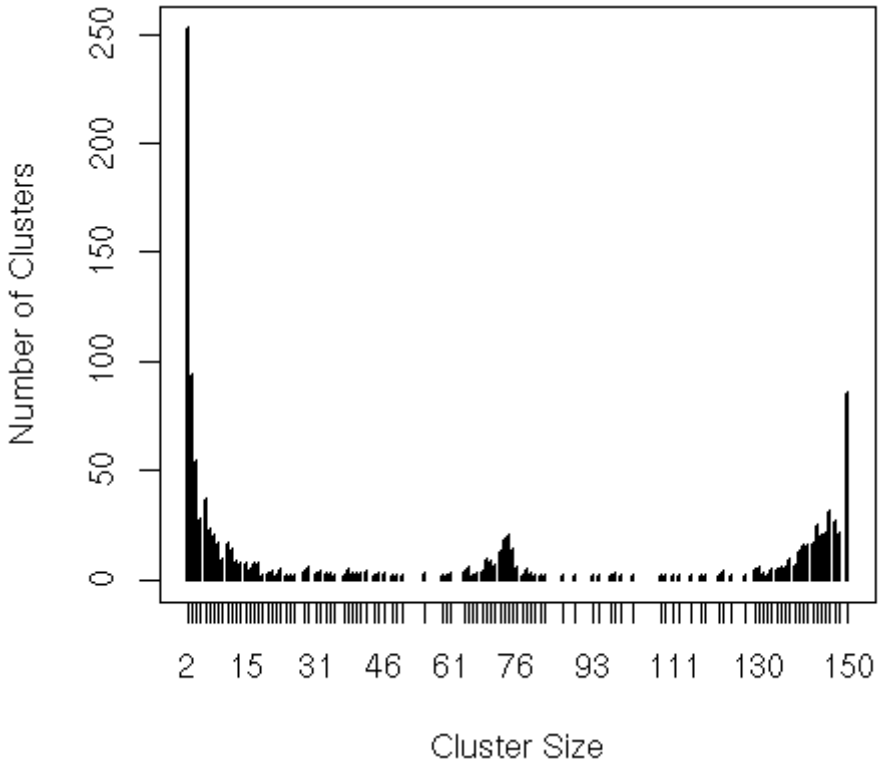

Figure 6: Histogram of Cluster Sizes for 63476 Ports Assigned to 1127 Clusters

Figure 7 shows cluster size as a function of the median numerical value of the port in the cluster. This graph illuminates some trends in network behavior by port. Reserved ports (ports below 1024) do not tend to correlate closely with their neighbors. Only 15% of reserved ports were clustered within the 0.05 threshold, with only three clusters consisting of five or more ports. Two clusters of 8 ports appear, one within the 450-600 range (452, 462, 485, 502, 506, 507, 575, 579), and one within the 600-700 range (623, 634, 639, 656, 662, 671, 678, 682). In comparison, 98% of Microsoft Windows ephemeral ports (1025 to 5000) were clustered with at least one neighbor, with only 73 ports (1.8%) belonging to clusters with between two and five ports. The group of ports between 5001 and 10000 show wider variation in cluster size than the ports typically used for both Windows client connection initiations, as well as the high-valued ephemeral ports used in UNIX connections. A total of 312 ports (6.8%) belong to clusters with between two and five ports; the maximum cluster size within the unassigned range is 118 ports. High-valued ports appear most highly correlated with their neighbors in general; 98.5% of ports were clustered with at least one neighbor, with a total of only 609 ports (1.1%) belonging to clusters with between two and five ports, and 38997 ports belonging to clusters with 140 ports or higher.

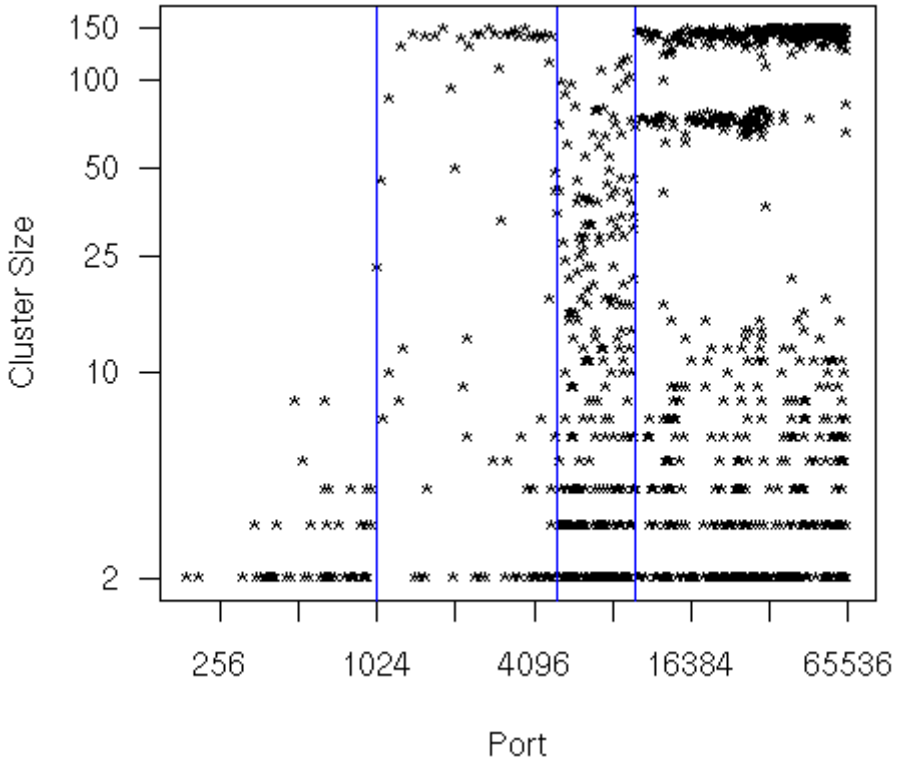

Figure 7: Cluster Size by Median Port Value per Cluster, Shown on a Log_2 Scale
Few reserved ports (0-1024) show correlated activity strong enough for clustering. Windows ephemeral ports (1025-5000) show similar cluster sizes to Unix ephemeral ports (10001-65535), with the exception of a group of clusters of size ~75 in the 10001-32768 range. Unassigned ephemeral ports (5001-10000) show much higher variation in cluster size.

Visualizations such as Figure 6 and Figure 7 may contain both global features that may be expected in any implementation of port clustering and features specific to a monitored network. For example, smaller cluster sizes can indicate a prevalence of ports in that region that receive a substantial amount of application-specific traffic. Reserved ports tend to cluster only in small

groups, if at all. On the other hand, the larger ephemeral ports (10000 to 65536) seem to have many more similar adjacent ports. This pattern may also arise from a smaller set of internal UNIX hosts that use those ports, as opposed to a large group of Windows hosts. These figures can be used as a high-level baseline of standard activity. Changes in cluster sizes as the algorithm is re-run over time may be indicative of changes in overall port usage or in internal network structure.

4.3.2 Frequency of Alerts

Port-specific means and variances (μ_j, σ_j^2) were estimated using the training data for every port correlated with at least one neighbor. Post-clustering variance adjustments were also computed using the training data, as noted in Appendix B, in order to produce Z-scores for both the training data (April 1 through April 7) and for the test data (April 8 through April 21). Figure 8 shows the number of cluster rejections by hour over time, flagged using the FDR method with a family error rate $\alpha = 0.01$.

Despite the post-variance modeling adjustments for time dependency, a distinct daily trend is seen in rejection rates, suggesting a model mis-specification error. Rates also increase with time, suggesting that model maintenance should include re-estimation of both cluster assignments and data-dependent parameters. Based on ordered p-values, the average hourly port rejection rate was 0.004, which corresponds to approximately 255 port rejections per hour out of the 63476 ports. On average, each cluster was rejected in 12% of hours during which it was tested. Nine clusters were flagged for over 50% of the 504 hours. This high cluster-specific volume can be attributed both to port surges and to model mis-specification.

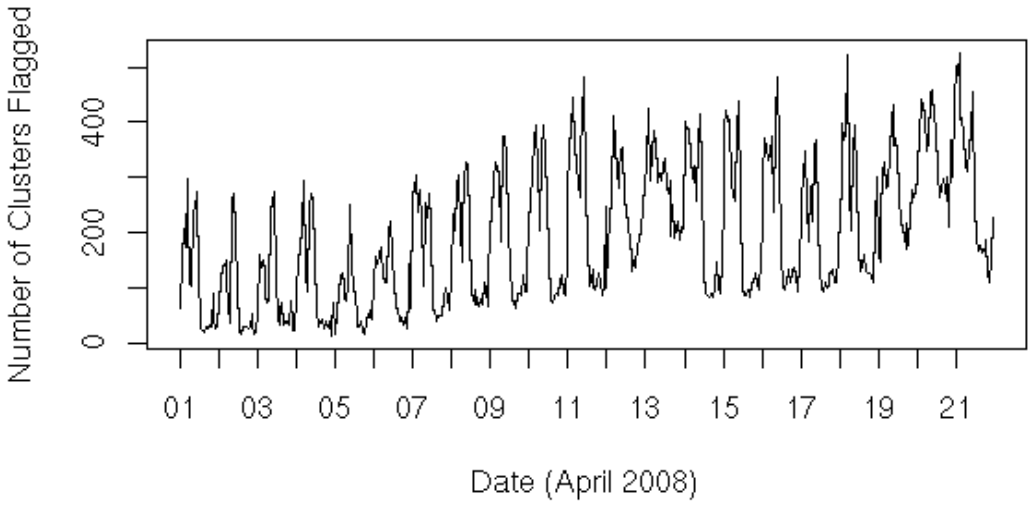

Figure 8: The Number of Cluster Rejections Over Time, Flagged Using a Family Error Rate $\alpha = 0.01$. The number of cluster rejections over time, flagged using a family error rate $\alpha = 0.01$. Despite the post-variance modeling adjustments for time dependency, a distinct daily trend is seen in rejection rates, suggesting a model mis-specification error. Rates also increase with time, suggesting that model maintenance should include re-estimation of both cluster assignments and data-dependent parameters.

We use a visualization called a *cluster summary graph* to examine clusters in more detail. Figure 9 shows an example. The top row is a time series plot of all Z-scores in the cluster. The ports with the ten highest Z-scores over the plotted interval are listed on the far left column, along with the

color of each port in the time series plot. The summary graph also displays several visual depictions of cluster health. The middle row is an indicator of the number of ports that were flagged as anomalous in the cluster, using the two-step alerting technique described in Section 3.5. A circle indicates that more than five ports in the cluster were flagged, a sign that the cluster may be unhealthy. The bottom row shows two diagnostic plots of Z-scores to determine the adherence to the normal model. A standard normal histogram displays a bell shape with almost all mass concentrated between ± 4 standard deviations. The QQ-plot indicates deviations from the normal model when the ordered percentiles deviate from the marked blue line. Heavy tails cause an "S" shape—first dipping below, then veering above the blue line. High numbers of rejections, as well as S-shaped QQ-plots, are indications of poor cluster health.

The cluster in Figure 9 is generally healthy with several anomalies. Time series plots of Z-scores do not show daily trend patterns. Obvious spikes in activity during week 3 are flagged by the FDR-based cluster rejection step, with generally fewer than three ports flagged using ordered p-values. The histogram and QQ-plot show adherence with standard normal tails, barring outliers from the port surges. The 10-hour surge on April 16 corresponds to a scan of port 5110 over several class B networks by multiple external hosts. Although activity post-scan increased on the morning of April 17, in this instance there was no internet-wide outbreak. Port 5110 is associated with the ProRat (remote administration tool) Trojan that runs a backdoor on that port. Port 5110 is also the default port for command line and GUI tools to connect to the IBM Tivoli Storage Productivity Center for Replication server. The activity we observed on port 5110 could be associated with either of these systems or could be related to some unknown other system.

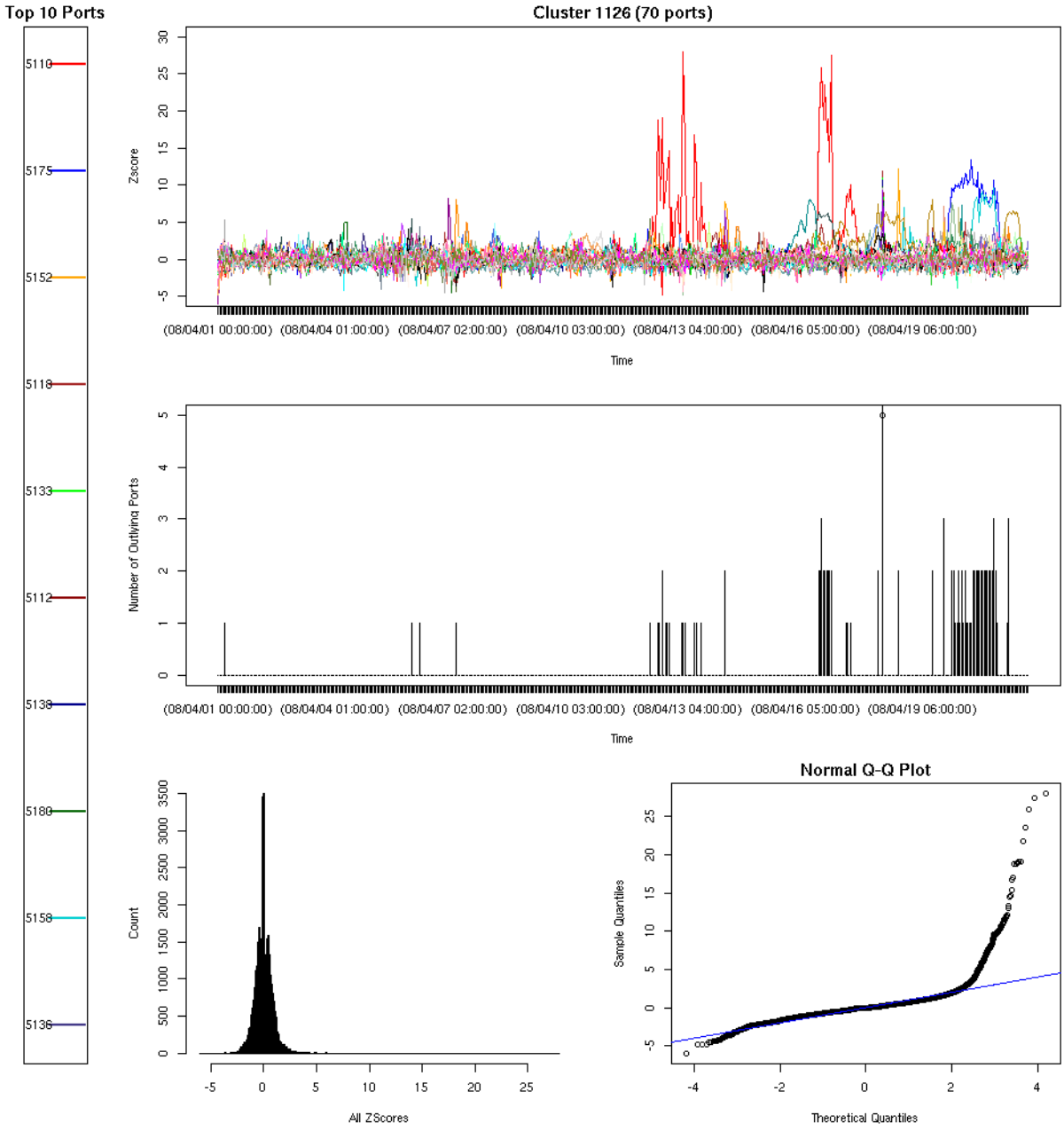

Figure 9: A Healthy Cluster with Several Anomalies

Time series plots of Z-scores (top row) show little daily trend patterns. Obvious spikes in activity during week 3 are flagged by the FDR-based cluster rejection step (middle row), with generally fewer than three ports flagged using ordered p-values. The histogram and QQ-plot (bottom row) show adherence with standard normal tails, barring outliers from the port surges.

The cluster in Figure 10 shows signs of poor health along with anomalies. Both the Z-score time series and number of rejected ports show evidence of daily trend as well as multiple ports triggering anomalies. The histogram and QQ-plot show slight skewness toward heavy upper tails. The cluster appears to gain anomalies over the course of the two-week testing period, an indication for the need of re-clustering with time.

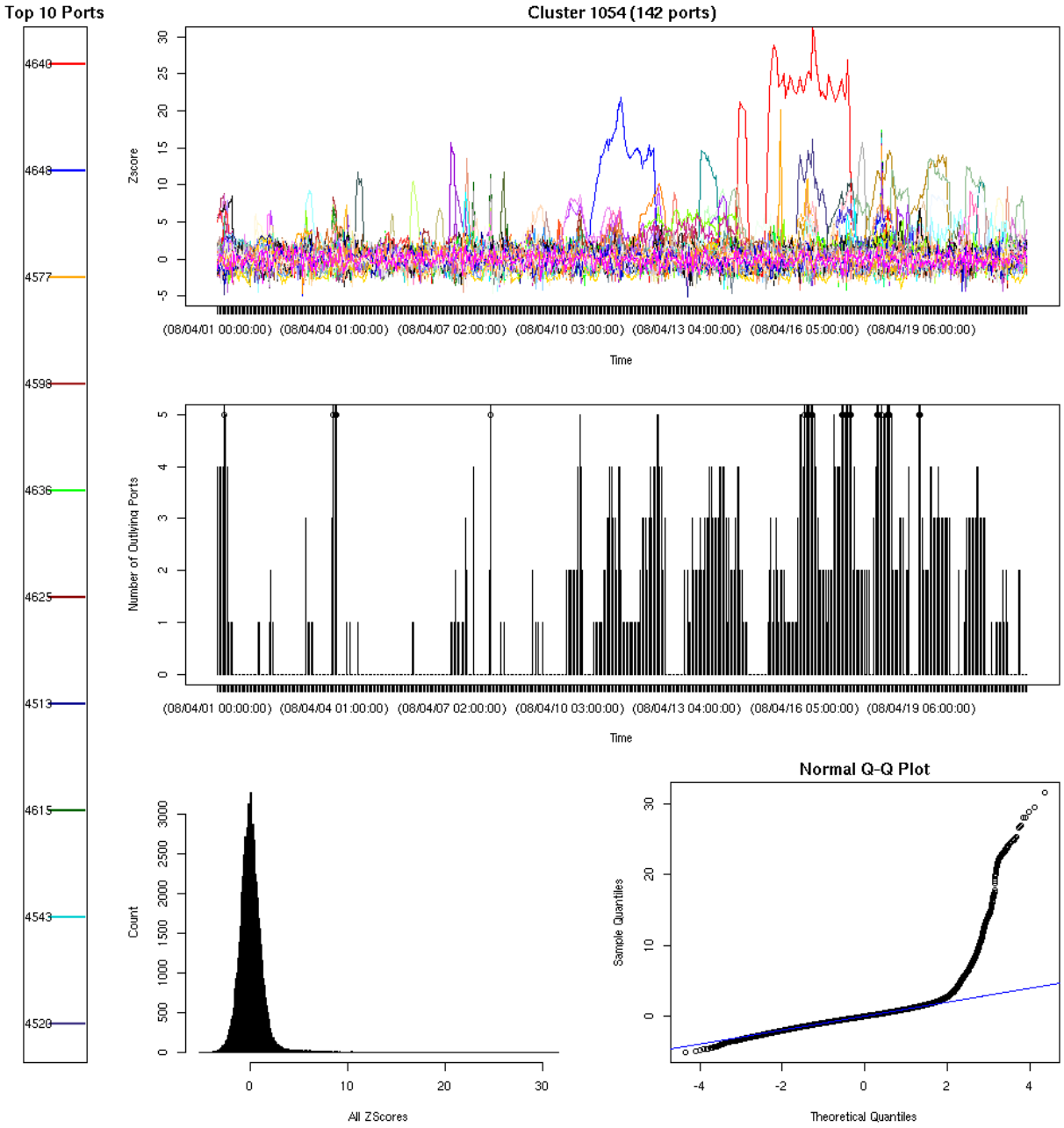

Figure 10: A Cluster Showing Signs of Poor Health along with Anomalies

Both the Z-score time series and number of ports rejected show evidence of daily trend as well as multiple anomalies. The histogram and QQ-plot show slight skewness toward heavy upper tails. The cluster appears to gain anomalies over the course of the two-week testing period, an indication for the need of re-clustering with time.

The cluster in Figure 11 is unhealthy; despite few visible surges or spikes in the Z-scores, this cluster was flagged in 54% of hours in which it was tested. Daily trends can be seen in the Z-score time series plots, with few obvious outliers. However, ordered p-values often flag multiple ports in a single hour. The histogram and normal QQ-plot suggest that Z-scores in this cluster have much heavier tails than the standard normal distribution. This model mis-specification leads to high rejection rates and false positives.

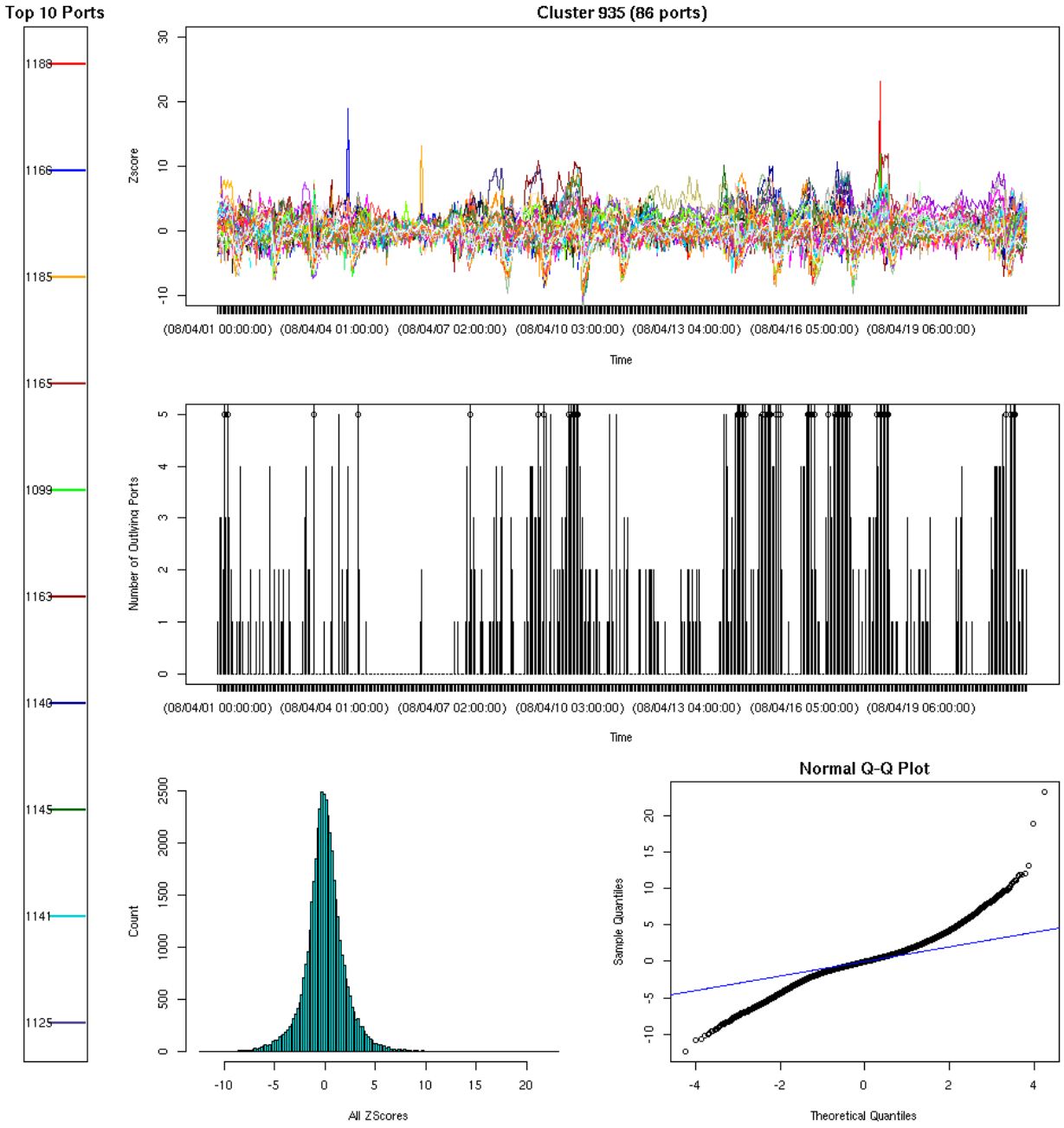

Figure 11: An Unhealthy Cluster Rejected in 54% of the 504 Hours Using the FDR Method
Daily trends can be seen in the Z-score time series plots, with few obvious outliers. However, ordered p-values often flag multiple ports in a single hour. The histogram and Normal QQ-plot suggest that the Z-scores have much heavier tails than a Normal distribution. This model mis-specification leads to high rejection rates and false positives.

4.3.3 Alert Classification and Evaluation

We took a stratified sample of 351 Z-scores with values greater than 5 across 17 strata used for post-processing variance adjustments (see Appendix B for details). Each sampled Z-score was

then expanded into a temporal event, based on the associated port's activity in adjacent hours. The port was marked as *surging* for the duration of contiguous hours that both spanned the sampled point and that had a Z-score greater than 5. A panel of experts on our team investigated each event to determine the source of the anomaly. We used this sample evaluation to estimate the percentage of all flagged events belonging to each category for the duration of the study. The results are presented in Table 3.

Table 3: Percentages of Categories of Events Discovered Using Port Clustering and Two-Stage Statistical Outlier Detection
The average duration of events ($\overline{D}$) is reported in hours, with associated standard deviations.

Type	% of Events	$\overline{D}$ (hours)	SD($\overline{D}$)
P2P Ghosting	41.0	19.8	1.6
Keep-alive	14.0	9.0	2.8
HTTPS (port 443) activity	12.0	7.6	1.4
FTP/SMTP activity	5.2	8.6	2.8
Misconfiguration/Backscatter	4.3	6.2	1.3
Connection Attempt	4.3	7.6	3.3
User Non-Web activity	3.9	22.0	11.9
SYN flood	3.6	9.0	4.4
Beaconing (Non-443)	3.6	20.5	5.0
Large Data Transfer	3.1	16.0	7.0
Scan activity	2.8	6.5	2.9
Statistical False Positive	0.8	1.2	0.2
File Transfer	0.6	3.0	2.0

The most common event type (an estimated 41% of all flagged events) is a specific kind of peer-to-peer *ghosting* activity. This anomaly is characterized by an inactive internal IP address that suddenly starts receiving variable-sized connection requests from multiple external IP addresses. The activity initiates suddenly, and can persist for long periods of time. This activity is observed when the internal IP address is mistakenly listed as a valid peer in a peer-to-peer network.

The majority of anomalous events flagged in the system can be classified as benign network activity. Session keep-alives (14%), HTTPS activity (12%) and FTP or SMTP activity (5.2%) represent user-initiated behavior that, while port-specific, generally does not require a security alert. Misconfigurations or backscatter (4.3%) and repeated connection attempts (4.3%) are attributable to repetitive machine activity.

An estimated 13% of flagged events are composed of suspicious events such as beaconing (keep-alives on non-standard ports), SYN floods, scanning, and large data transfers. For example, Figure

12 shows a visualization of a large multi-port scan that was flagged across several clusters. Each *comb plot* in the figure represents a successive hour and displays a port number on the X axis and a Z-score on the Y axis. Only Z-scores with values greater than 10 are plotted. Blue dashed lines correspond to ports: 1024, 500, 10000, and 32768. Activity first occurs at 4 a.m. on low-valued Windows server ports (1024 to approximately 1500) and low-valued UNIX ephemeral ports (10000 to approximately 15500). By 5 a.m., the activity migrates across Windows server ports, ephemeral unassigned ports (5000-10000), and a cross-section of the remaining UNIX ephemeral ports. There are indications of reserved port activity (0-1024) for the 6 a.m. hour that may also be associated with the scan.

Statistical false positives (an estimated 0.8% of all flagged events) are events that appear to show no port-specific behavior upon examination. The rate of statistical false positives appears consistent with the FDR rate that was set for the two-stage hierarchical outlier detection. But the failure of the algorithm to detect actionable anomalies is due less to the statistical methods and more to variable selection and data pre-processing. The simple flow volumes that were used as inputs for port correlation and anomaly detection also contained a large amount of benign port-specific activity.

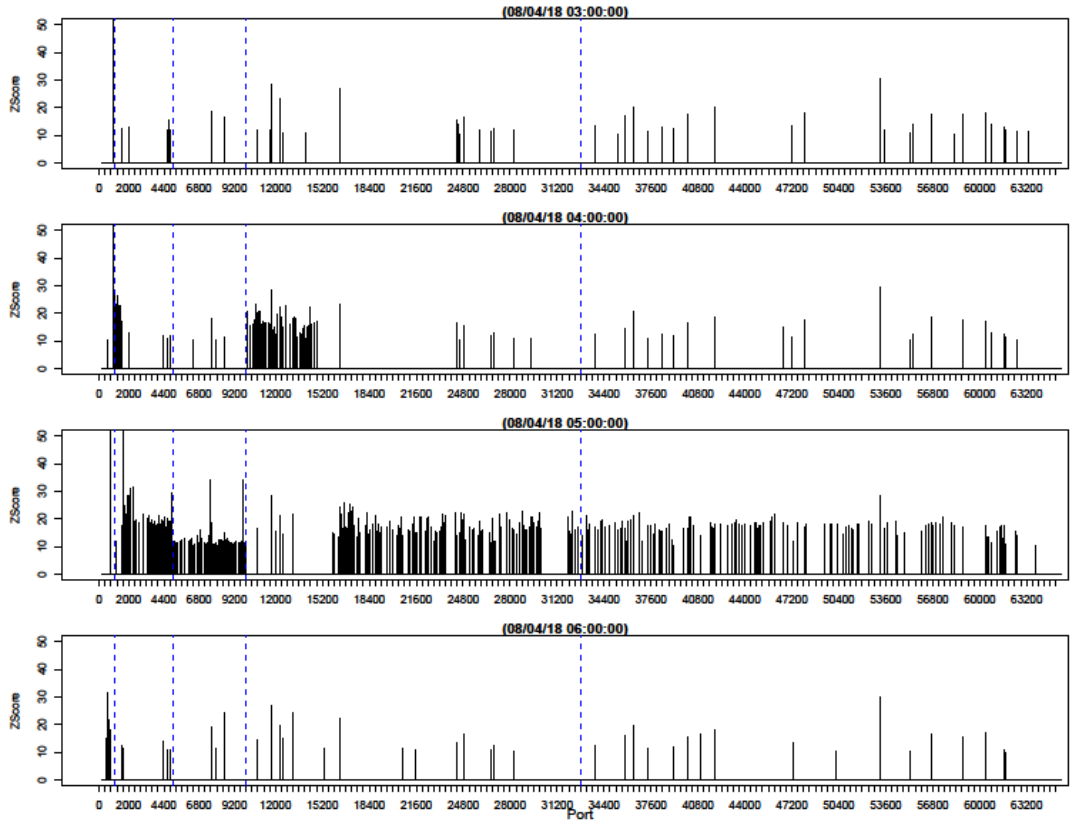

Figure 12: Sequential Comb Plots Show the Result of a Two-Stage Multi-Port Scan
For visual reference, blue dashed lines correspond to ports: 1024, 500, 10000, and 32768. Activity first occurs at 4 a.m. on low-valued windows server ports (1024 to approximately 1500) and low-valued UNIX ephemeral ports (10000 to approximately 15500). By 5 a.m., the

activity migrates across Windows server ports, ephemeral unassigned ports (5000 to10000), and a cross-section of the remaining UNIX ephemeral ports. There are indications of reserved port activity (0 to1024) for the 6 a.m. hour that may also be associated with the scan.

4.4 Future Work: Variable Selection and Context

One way to increase the utility of the port-specific anomaly detector is to use more context-specific volumetric data, as opposed to hourly flow counts per port. For example, a flow count for a single port can be abnormally high due to any of the following events occurring on that port:

- One external host sends flows to one internal host.
- One external host sends flows to multiple internal hosts.
- Multiple external hosts send flows to one internal host.
- Multiple external hosts send flows to multiple internal hosts.

Event 1 is associated with misconfigurations, backscatter, and machine-initiated connection attempts, which are not high-priority actionable security events. Event 2 is the hallmark of a *lone, loud scanner*—an IP address that does not use any proxies or other obfuscation when performing port scans. These blatant scanners are generally easy to find in the course of day-to-day network monitoring with, for example, top-N lists. Event 3 is typified by the peer-to-peer ghosting activity that, while anomalous, is a low-priority actionable event. Event 4 is the event most likely to precede a port-specific surge due to increased interest in a recent vulnerability. As news of the vulnerability spreads among hacker communities, increasing numbers of external hosts should be observed performing scans of multiple internal hosts on the network.

To track port-specific anomalies due to new exploits or vulnerability releases, a more informative measure might be the number of unique external IP addresses scanning each port per hour, with an associated summary of the number of unique destination ports scanned per IP address. Restricting IP addresses to only those marked as scanners by a blacklist, IDS tool, or prior analysis also gives the measured data a smaller scope and context for flagging volumetric anomalies.

To explore this new data, we used the combined Thresholded Random Walk and MISSILE algorithms, built into the SiLK rwscan tool[3], to flag a set of scanners between the dates of November 1 and December 31, 2008. In six-hour intervals, we recorded the number of unique external IP addresses scanning a large network, as well as a 6-tuple summary (min, max, median, 75th percentile, 97.5th percentile, and log-mean) of the number of unique internal addresses scanned by each source IP address. These seven time series were recorded for each of the 65536 unique ports. As a first step to explore surging ports, we performed linear regressions of unique source IP (log-scaled) over time and ranked ports by the steepness (either increasing or decreasing) of the resulting slope estimate. Figure 13 shows some examples of highly ranked ports, plotted on a log scale. The 97.5th percentile of the number of unique destinations scanned is represented on the log scale by color on the graph. Green colors indicate that the majority of IP addresses in the six-hour window scanned fewer than 64 internal hosts. Yellow and red indicate a log-scale increase in this number.

[3] Available at http://tools.netsa.cert.org/silk/rwscan.html.

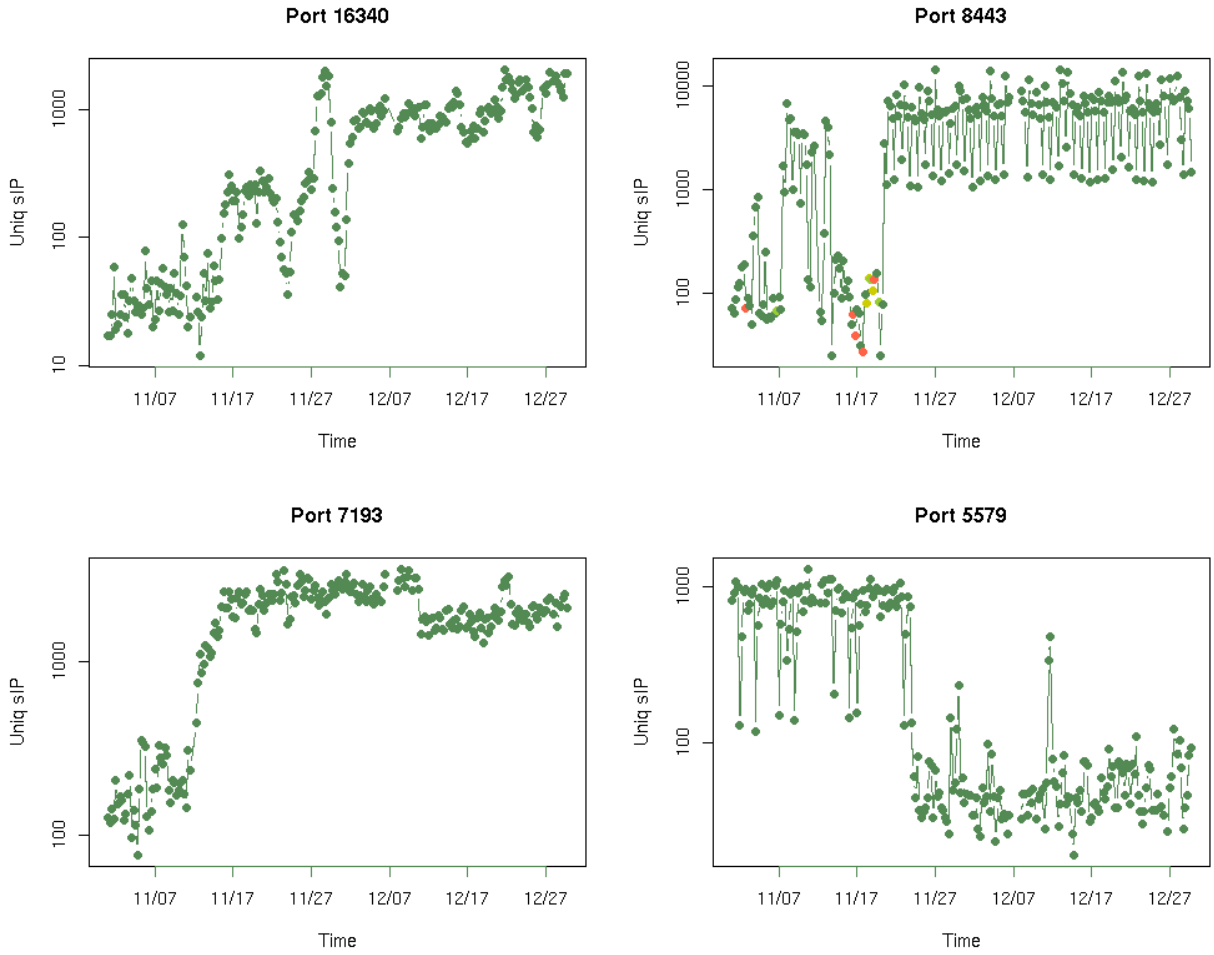

Figure 13: Volume of Unique Sources Scanning Several Different Destination Ports

In the case exemplified by port 16340, the volume of unique scanners increases gradually over time, as we hypothesized would occur with a vulnerability announcement. However in other cases, such as ports 8443 and 7193, the number of scanning IP addresses increases rapidly by orders of magnitude. Rather than a continued increase, the high-volume activity stabilizes, often over a period of weeks or months. Rapid declines, as exemplified by port 5579, are also evident. We hypothesize that this activity pattern is due to botnets. The colors of the points represent the number of addresses scanned by the majority of source IP addresses, with green indicating few addresses scanned per source, and red and yellow colors indicating higher numbers. Only in the case of port 8443, during the week of November 17, does it appear that incoming sources also scan large parts of the internal network.

This kind of botnet activity can lead to false positives in an alerting system looking for new port-based vulnerabilities; none of these incidents were associated with an outbreak. We hypothesize that scanning would not only increase, but that scanners would become bolder in scope, targeting larger areas for scanning of the vulnerable port, as opposed to targeted surgical attacks. We can also see evidence of correlation among ports with this data as well. For example, Figure 14 shows time series plots for ports 7193 and 8516 that display strikingly similar patterns, perhaps

indicating a strong overlap in the external hosts that scanned both ports. Such fingerprinting can be used not only as a basis to cluster similar ports but also to track botnets and related scanners as they cycle among hosts.

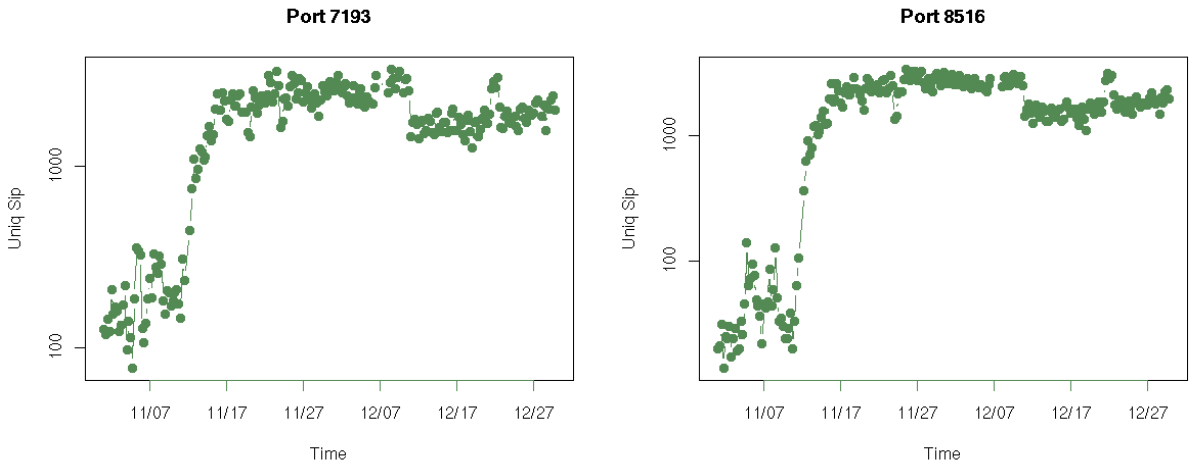

Figure 14: Time Series Plots Displaying Similar Patterns

Similar patterns in port scanning activity suggest coordination among unique sources and motivate a cluster-based approach for discovering botnet associations.

5 Discussion

The port-specific anomaly detection methodology, implementation, and evaluation discussed in this report provide a case study in applying proof-of-concept research techniques to a large-scale, operational network security environment. This report details the steps our research team took for scaling a novel volumetric anomaly detection method (clustering and median-based trend correction) into an automated, adaptive system tasked to monitor thousands of internal resources simultaneously. Section 5.1 details the lessons learned specific to the port-based anomaly detection methodology. Section 5.2 provides some broader recommendations for large-scale volumetric anomaly detection.

5.1 Lessons Learned

5.1.1 Clustering

The main reasons for using correlation-based clustering as the method for modeling trend were

- to categorize related ports into groups in order to organize and present information to the analyst
- to take advantage of a state-free, simple median correction for obtaining residuals on which to base thresholds
- to reduce the number of hypothesis tests needed for initial flagging of events

Although the original research provided a proof of concept that highlighted several empirical clusters, scaling the port clustering algorithm to automatically correlate and group 65536 ports was a substantial challenge. To relax the requirement of looking only at small sets of adjacent pairs, agglomerative hierarchical clustering, which requires full pairwise comparisons, can be replaced by a faster, adaptive method such as BIRCH [7]. But finding useful thresholds for both pairwise MCD correlation and for defining clusters requires an extensive exploratory analysis. Parameter settings generally will not translate well between differently configured and sized networks, or between different measures of activity (for example, byte counts vs. flow counts vs. unique host counts). It is also clear from the three-week data set that clusters based on flow volumes age rapidly, and that re-clustering would need to be performed often in an operational environment. Normal user activity can initiate large deviations from established trends, which calls to question examining changes to clusters over time as an anomaly detection method.

In our case study, the simplicity of state-free, median-corrected residuals also did not scale well to an automated anomaly detection system. Median-corrected time series were not sufficiently stationary and uniform for removing temporal trends and applying universal thresholding to discover statistical outliers. Distributions of median-corrected port series varied with time of day, cluster size, and cluster median value. We required significant post-processing of median-corrected port volumes to account for these sources of variation, with limited success. The log-scale transformation was not sufficient to correct for changes in both trend and variation with time, and the post-processed Z-scores also continued to suffer from temporal and volumetric effects, as well as individual differences among ports. This suggests that port-specific parameters (that is, keeping state per port) are still required in an operational setting, even when ports are

grouped into clusters. Although the cluster-based method did require significantly fewer hypothesis tests per iteration for flagging outliers, the advantage came at a much higher cost of complexity in the baseline modeling than we anticipated.

5.1.2 Parameters

To transform median-corrected series into Z-scores, a simple model was chosen to account for residual trend. Once the cluster median value was subtracted, any remaining trend was modeled only by time-independent baseline means μ_j^c for each clustered port. Port-specific variance, cluster size, median value, and temporal effects were all modeled as variance components. But daily trends in rejection rates in both training and test data suggest that the baseline trend model should be more flexible. For example, a linear model with global, cluster-specific and port-specific coefficients can be implemented. We anticipate that increasing flexibility in baseline trend modeling will help to alleviate the temporal effects in rejection rates. Following Lambert and Liu [**8**], autocorrelation functions of trend-corrected residuals can be examined to determine if the resulting residuals are time-independent.

5.1.3 Data Collection and Alerts

For the hourly count data used in this analysis, it appears that training the anomaly detection system on one week of data is insufficient for generalizing cluster values and parameter estimates beyond a few days. Baseline studies of network behavior, using historical data collected over many weeks, will help to inform trend models.

Though flow counts were filtered to reduce the effect of backscatter, the resulting volumes still displayed surges due to benign network activity. Adapting the method to count unique source IP addresses and unique destination addresses restricts the scope of the port-specific anomaly detector to events including multiple external addresses and multiple internal addresses. This provides a more targeted scope for an anomaly detection system, and preliminary analysis indicates that the reduced scope may produce more actionable anomalies. The method can also be adapted to calculate Z-scores based on outlying points in multivariate distributions. This kind of adaptation would produce alerts based on information from more than one measurement, simultaneously.

Statistical false positives appear to be managed well using the FDR method. To detect surges, however, it may be more useful to track average Z-scores by port over time. A similar technique is also suggested by Lambert and Liu [**8**], who apply exponentially weighted moving average models to Z-scores in a control chart approach. Smoothing Z-scores before flagging anomalies can help to reduce false positives, but corrections for multiple hypothesis testing will still be required when many thousands of network resources are monitored simultaneously.

5.2 Recommendations

In a large-scale operational environment, even a simple statistical model must often be conceptualized as an *expert system*. Scaling a proof-of-concept idea to an implemented system for monitoring thousands of assets requires attention to more than a simple predictive model. A proof-of-concept methodology that neither addresses nor evaluates diversity of assets, scaled false positive rates, model evolution, and diagnostics and maintenance after implementation presents many challenges and unknowns to the applied researcher tasked with turning that concept into a

useful tool. Based on our evaluation and analysis of the port clustering and alerting algorithm, we recommend the following for large-scale automated volumetric anomaly detection:

1. **Choose variables in context**. Simple measurements may be useful for very specific trend analysis, for example, on a single-host level. But as the scope of the network monitor increases, and as metrics aggregate over many nodes, interpretable and actionable anomaly detection should become more targeted in scope, with appropriate metrics chosen to give a useful volumetric view that highlights a specific type of anomaly. Multivariate methods can incorporate information from several metrics simultaneously and may be more useful than the univariate methods often cited in the literature.
2. **Carefully model both trend and residuals.** Statistical models describe both trend and residual variation once trend is removed. In anomaly detection, the residual variation is important to model accurately, especially for describing which extreme values are nonetheless typical (*tails* of the distribution) and which are truly anomalous. It is most important that the model for these residual extreme values is accurate across the population of assets that are being monitored, before universal thresholding can be used. Building models to monitor thousands of assets requires extensive historical reference data, diagnostics designed to validate the model at extreme values, and an exploratory pre-implementation phase focused on ranking goodness of fit across network assets that will be monitored.
3. **Correct for multiple hypothesis testing.** When faced with the possibility of thousands of statistical tests to determine alerts, a method that controls for multiple hypothesis testing, such as Benjamini and Hochberg's [**6**] method for controlling the False Discovery Rate (the FDR method), or the FDR method coupled with control-chart methods, should be used to ensure that any system conform to pre-determined type I error rates. The FDR method is especially appealing because the algorithm is simple and easy to implement on a large scale.
4. **Provide adaptive, interpretable methods for model diagnostics and evolution.** Anomaly detection systems based on statistical models should include methods for diagnosing model misfit and for updating model parameters. Inline methods (such as exponential moving averages) can be used, but even these methods may not be robust enough to adapt the model adequately over time. Simulations and application to historical data can be used to devise schedules for model maintenance. Analysts and operators that rely on any large-scale anomaly detection system should be trained on the interpretation of model diagnostics and on the procedures that should be taken when diagnostics indicate model mis-specification.

Appendix A The Minimum Covariance Determinant (MCD) Algorithm

For a cloud of N data points in P-dimensional space, with $P < N$, the MCD algorithm constructs a robust measure of center and spread of the cloud. For non-robust methods, the center is measured by the P-dimensional mean vector T, and spread is measured with a $P \times P$ covariance matrix Σ, where the ij-th element of Σ stores the covariance measured for dimensions i and j. The MCD algorithm takes as an argument a subset size $H < N$ and computes T_* and Σ_*, where the mean and covariance measures are taken not over all data points, but over the H data points for which the resulting covariance matrix has a minimum determinant over all possible subsets of size H. This provides a robust estimate of center and spread that is not strongly influenced by outliers.

Despite having many nice statistical properties, the MCD estimator was not used in practice (it was supplanted by a method called MVE or "minimum volume ellipsoid") because it was slow to compute. However, Rousseeuw and van Dreisen [**3**] devised a fast method for computing T_* and Σ_*, and now maintain that MCD is the preferred method for robust center and spread measures as opposed to MVE.

The workhorse of the MCD algorithm is the *C-step* or "concentration step." Starting with a center T_0 and spread Σ_0 calculated using a subset of arbitrary size, the C-step first requires computation of the Mahalanobis distance of each data point x_i to T_0 with respect to Σ_0, given by

$$d_i = d(x_i, T_0, \Sigma_0) = \sqrt{(x_i - T_0)' \Sigma_0^{-1} (x_i - T_0)}$$

The data points x_i are sorted according to their distances d_i, and the subset of H points with smallest distances are used to obtain a new center T_1 and spread Σ_1, for which $det(\Sigma_1) \leq det(\Sigma_0)$. Because covariance matrices are positive-definite, their determinants are bounded below by 0, and so repeated applications of C-steps are assured to converge.

The method for finding Σ_* does not compute all possible subsets, but instead proceeds as follows:

1. Repeat ν times:
   a. Construct an initial T_0, Σ_0 using a randomly chosen subset of $P + 1$ points for which $det(\Sigma_0) > 0$.
   b. Carry out two C-steps.
2. For the 10 results in Step 1 with lowest determinant, carry out C-steps until convergence.
3. Report T_* and Σ_* as the solution with the smallest observed determinant.

Rouseeuw and van Dreisen suggest $\nu = 500$ as a reasonable number with which to seed the algorithm.

Appendix B Port-Specific Models and Post-Clustering Variance Adjustments

The model for each clustered port includes a port-specific baseline mean μ_j^c and standard deviation σ_j^c. These are estimated using the median-subtracted time series $\{Y_j^c - \tilde{Y}^c\}$. To calculate the trimmed mean for N observations in a time series, let $0 \leq \gamma < 1$ be a trim proportion. To estimate the baseline mean, calculate

$$\hat{\mu}_j^c = \frac{1}{N^*} \sum_{t=\lceil (1-\gamma/2)N \rceil}^{\left\lfloor \frac{\gamma N}{2} \right\rfloor} \left(Y_j^c - \tilde{Y}^c\right)_{(t)}$$

where $N^* = \left\lfloor \frac{\gamma N}{2} \right\rfloor - \lceil (1-\gamma/2)N \rceil + 1$, and the index (t) indicates the t-th ordered value of the N observed values in the time series. This is a trimmed mean taken across the middle γ proportion of the median-subtracted port volumes. To estimate the baseline standard deviation, calculate

$$\hat{\sigma}_j^c = \sqrt{\frac{1}{N^* - 1} \sum_{t=\lceil (1-\gamma/2)N \rceil}^{\left\lfloor \frac{\gamma N}{2} \right\rfloor} \left[\left(Y_j^c - \tilde{Y}^c\right)_{(t)} - \hat{\mu}_j^c \right]^2}.$$

The use of the trimmed mean and standard deviation is to account for noisy data collection that is contaminated by the same kinds of events that would initiate alerts for surges. The estimated values $\hat{\mu}_j^c$ and $\hat{\sigma}_j^c$ are plugged in to the calculation of S_{jt}^c

$$S_{jt}^c = \frac{\left(Y_{jt}^c - \tilde{Y}_t^c\right) - \hat{\mu}_j^c}{\hat{\sigma}_j^c}$$

Figure 15 shows the median-corrected series S_{jt}^c for the training set (April 1 through April 7), plotted against general port volume across several different strata of time and cluster size. In each sub-plot, the y-axis is S_{jt}^c, and the x-axis is the log-scaled cluster median that was subtracted at time t to obtain S_{jt}^c. The names in each plot title correspond to the following variables:

- Cluster group (CG)
  - CG=1: Cluster size < 20 ports
  - CG=2: Cluster size between 21 and 50 ports
  - CG=3: Cluster size between 51 and 100 ports
  - CG=4: Cluster size between 101 and 150 ports
- Peak hours (PK)
  - PK=0: Relative time is between 8 p.m. and 8 a.m.
  - PK=1: Relative time is between 8 a.m. and 8 p.m.
- Workday (WK)
  - WK=0: Day of the week is a Saturday or Sunday
  - WK=1: Day of the week is Monday through Friday

The red line in each graph corresponds to a cluster median value of 20. We note from this diagnostic that despite scaling S_{jt}^{c} by a port-specific standard deviation, the variation changes with port volume, time, and cluster size. Starting with a median value of approximately 20, variance decreases with volume across almost all strata, leading to the cone shape in S_{jt}^{c} as the median value increases. Variance seems stable but large across all strata when the cluster median is 20 or less. The cone effect also seems more pronounced in the second and third columns corresponding to weekdays, and less pronounced for the first row corresponding to clusters of 20 or fewer ports.

We use the diagnostic plots in Figure 15 to heuristically group residuals for modeling variability in terms of median. For any value S_{jt}^{c} calculated when the median $\tilde{Y}_{t}^{c} > 20$, we scale it by a factor σ_{jt}^{m} according to the following model:

1. $m_{t}^{c} = \left\lfloor \frac{\tilde{Y}_{t}^{c}}{10} \right\rfloor - 1$
2. $\log\left[(\sigma^{2})_{jt}^{m}\right] = \theta_{0} + \theta_{1} \log m_{t}^{c} + \theta_{2} (\log m_{t}^{c})^{2}$

The expression for m_{t}^{c} bins median values into groups of 10 (for example, 21-30, 31-40, 41-50, etc.), and assigns each bin a numeric value starting at 1. The coefficients θ_{0}, θ_{1}, and $\theta_{2}$are estimated independently for 8 subgroups resulting from all combinations of conditions for peak versus non-peak hours, weekdays versus weekends, and small cluster size (≤ 20 ports) versus large cluster size (21 to 150 ports).

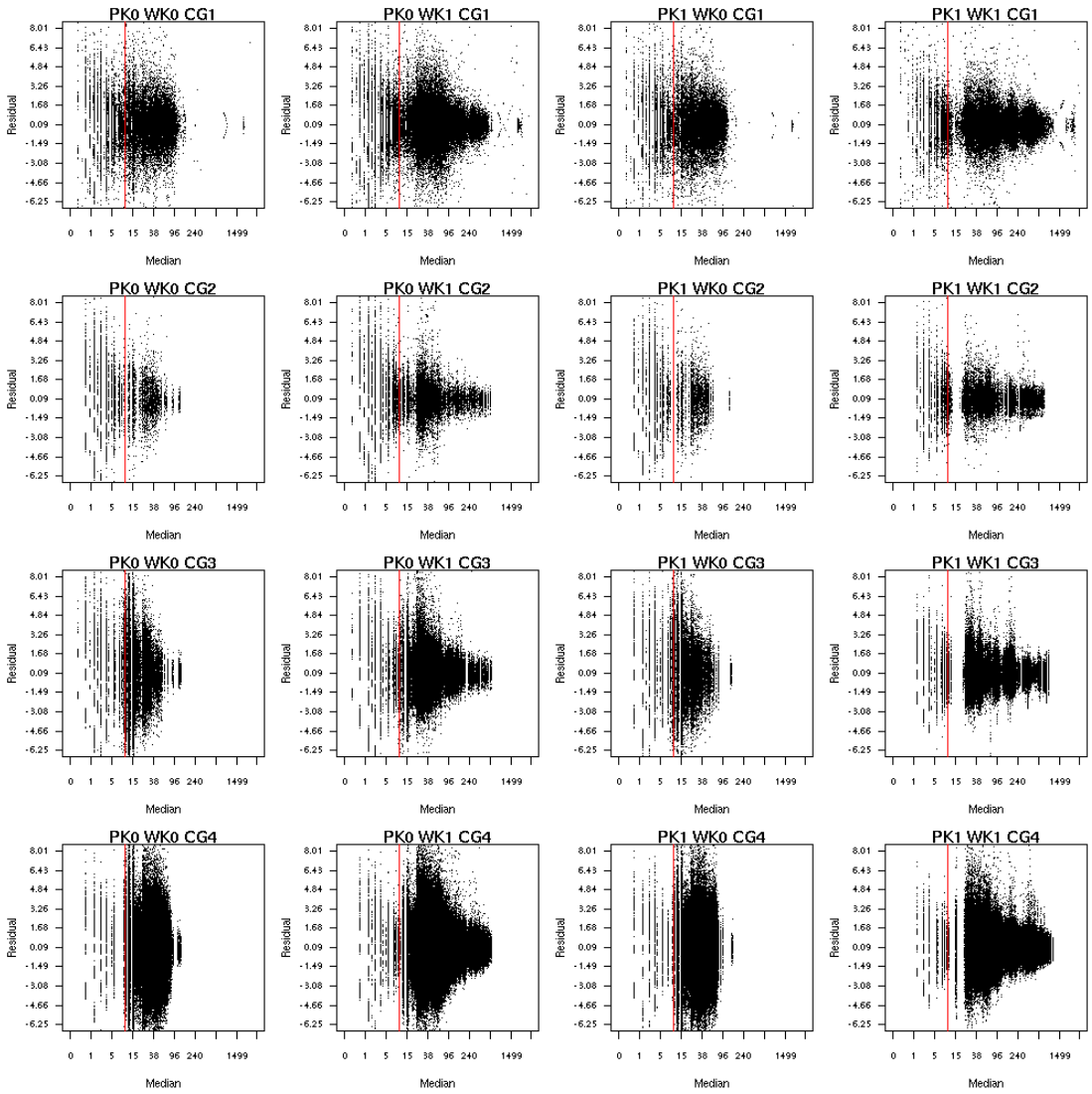

Figure 15: Residual Plots Vs. Cluster Median for 16 Different Strata

Residual plots versus cluster median for 16 different strata according to peak time of day (PK, 1 = 8am to 8pm, 0 = 8pm to 8am), day of week (WK, 0 = Mon-Fri, 1 = Sat or Sun) and cluster size (CG, 1 = 2 to 20 ports, 2 = 21 to 50 ports, 3 = 51 to 100 ports, 4 = 101-150 ports). Residuals appear to follow a universal pattern across all strata for median sizes below 20 (the red line in each graph). Residual variance universally decreases as the median size increases.

To estimate θ_0, θ_1, and θ_2 from the data, for each observed bin m with a total of n_m observations, we calculate

$$V_m = \frac{1}{n_m} \sum \left(S_{jt}^c\right)^2$$

We then regress $\log V_m$ on the bin value m with the model as expressed in Equation 2 on page 34. We use weighted least squares to account for unequal numbers of observations per bin. Figure 16 shows plots of the observed V_m by median on a log-log scale, with the associated regression lines. Table 4 lists the values of the parameters θ_0, θ_1, and θ_2 estimated from the data for each category.

The final scaling factor σ_{jt}^s, used to obtain the Z-score, takes one of seventeen values depending on the cluster size associated with port j, and the cluster median value and time of day at time t. A scaling factor for all residuals corresponding to a cluster median of under 20 is estimated using the standard variance calculation applied to training data from all 16 strata, with an observed median value of 20 or below (all points left of the red line in all strata in Figure 15). Scaling factors are estimated by strata for each of the remaining 16 strata using the standard variance calculation applied to the median-corrected values $\left(\sigma_{jt}^m\right)^{-1} S_{jt}^c$.

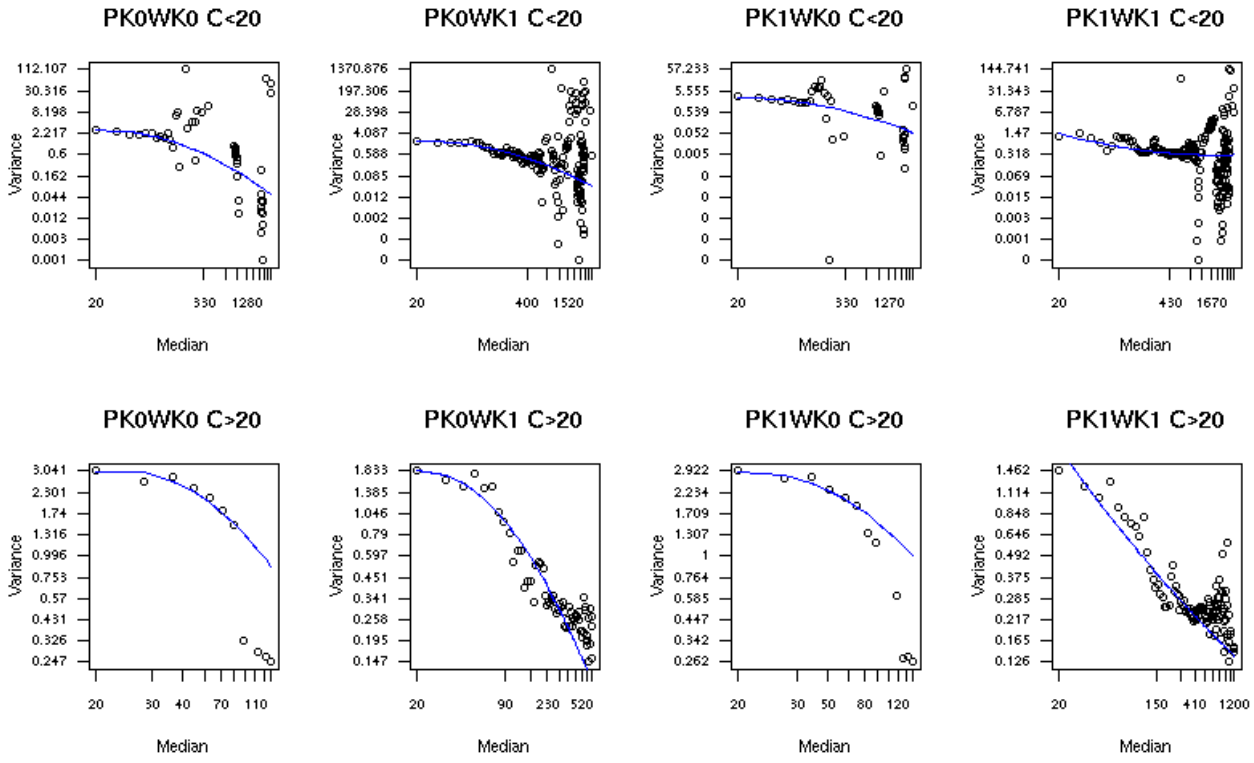

Figure 16: Weighted Quadratic Regressions of Variance

Weighted quadratic regressions of variance (estimated as average binned squared residuals) by median for 8 different strata, according to peak time, work day versus weekend, and cluster size.

Table 4: Estimated Parameters for the Regression Model of Variance Correction Values σ_{jt}^{m}

Hour of day	Day of week	Cluster size	θ_0	θ_1	θ_2
Peak hours	Weekday	≤ 20	0.9740	0.0411	-0.1313
Peak hours	Weekday	> 20	0.6839	0.0508	-0.1309
Peak hours	Weekend	≤ 20	1.0110	-0.0464	-0.1141
Peak hours	Weekend	> 20	0.3600	-0.5851	0.0553
Off-peak hours	Weekday	≤ 20	1.0907	0.2149	-0.2866
Off-peak hours	Weekday	> 20	0.6164	0.0336	-0.1697
Off-peak hours	Weekend	≤ 20	1.0600	0.0787	-0.1814
Off-peak hours	Weekend	> 20	0.6677	-0.6578	0.0212

Median	Cluster	Baseline variance σ^s Off-Peak Weekend	Off-Peak Weekday	Peak Weekend	Peak Weekday
$\tilde{Y} < 20$	All groups	2.9321 (all timing conditions)			
$\tilde{Y} \geq 20$	CG = 1	0.9421	0.9275	0.9521	0.9405
$\tilde{Y} \geq 20$	CG = 2	0.6729	0.5729	0.7475	0.4722
$\tilde{Y} \geq 20$	CG = 3	0.8209	0.6657	0.8771	0.7417
$\tilde{Y} \geq 20$	CG = 4	0.9715	1.0144	0.9656	1.0185

References

[1] P. J. Brockwell and R. A. Davis, *Introduction to Time Series and Forecasting*, 2nd ed. New York: Springer, 2002.

[2] J. McNutt and M. DeShon, "Correlations Between Quiescent Ports in Network Flow," Pittsburgh, research report, 2005.

[3] P. J. Rousseeuw and K. van Driesen, "A Fast Algorithm for the Minimum Covariance Determinant Estimator," *Technometrics*, vol. 41, pp. 212-223, 1999. [Online]. http://portal.acm.org/citation.cfm?id=331458

[4] J. O. Rawlings, S. G. Pantula, and D. A. Dickey, *Applied Regression Analysis*, 2nd ed. New York: Springer-Verlag, 1998.

[5] R. A. Johnson and D. W. Wichern, *Applied Multivariate Statistical Analysis*, 5th ed. Prentice Hall, 2001.

[6] 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*, vol. 57, pp. 298-300, 1995.

[7] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An Efficient Data Clustering Method for Very Large Databases," in *Proceedings of the ACM SIGMOD Conference*, Montreal, 1996, pp. 103-114.

[8] D. Lambert and C. Liu, "Adaptive Thresholds: Monitoring Streams of Network Counts," *Journal of the American Statistical Association*, vol. 101, pp. 78-88, 2006. [Online]. http://www.pearsonhighered.com/educator/product/Applied-Multivariate-Statistical-Analysis/9780131877153.page

[9] D. Lambert and C. Liu, "Adaptive Thresholds: Monitoring Streams of Network Counts," *Journal of the American Statistical Association*, vol. 101, pp. 78-88, 2006.

[10] P. J. Rousseeuw and K. van Driesen, "A Fast Algorithm for the Minimum Covariance Determinant Estimator," *Technometrics*, vol. 41, pp. 212-223, 1999.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
(Leave Blank)	May 2010	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
Identifying Anomalous Port-Specific Network Behavior	FA8721-05-C-0003

6. AUTHOR(S)
Rhiannon Weaver

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

8. PERFORMING ORGANIZATION REPORT NUMBER
CMU/SEI-2010-TR-010

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
HQ ESC/XPK
5 Eglin Street
Hanscom AFB, MA 01731-2116

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
ESC-TR-2010-010

11. SUPPLEMENTARY NOTES

12A DISTRIBUTION/AVAILABILITY STATEMENT	12B DISTRIBUTION CODE
Unclassified/Unlimited, DTIC, NTIS	

13. ABSTRACT (MAXIMUM 200 WORDS)

Increasing trends in traffic volume on specific ports may indicate new interest in a vulnerability associated with that port. This activity can be a precursor to internet-wide attacks. Port-specific behavior can also arise from stealthy applications that migrate to different ports in order to evade firewalls. But detecting this subtle activity among thousands of monitored ports requires careful statistical modeling as well as methods for controlling false positives. The analysis documented in this report is a large-scale application of statistical outlier detection for determining unusual port-specific network behavior. The method uses a robust correlation measure to cluster related ports and to control for the background baseline traffic trend. A scaled, median-corrected process, called a *Z-score*, is calculated for the hourly volume measurements for each port. The Z-score measures how unusual each port's behavior is in comparison with the rest of the ports in its cluster. The researchers discuss lessons learned from applying the method to the hourly count of incoming flow records for a carrier-class network over a period of three weeks.

14. SUBJECT TERMS	15. NUMBER OF PAGES
anomaly detection, TCP ports, large-scale analysis, clustering, vulnerability scans, statistical models, false discovery rate	46

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL