2012-09

Transport Traffic Analysis for Abusive Infrastructure Characterization

Nolan, Le E.

Monterey, California. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

TRANSPORT TRAFFIC ANALYSIS FOR ABUSIVE INFRASTRUCTURE CHARACTERIZATION

by

Le E. Nolan

September 2012

Thesis Advisor:	Robert Beverly
Second Reader:	Joel D. Young

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
19–9–2012	Master's Thesis	2010-06-01—2012-09-21

4. TITLE AND SUBTITLE

Transport Traffic Analysis for Abusive Infrastructure Characterization

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Le E. Nolan

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Postgraduate School
Monterey, CA 93943

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

N/A

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited

13. SUPPLEMENTARY NOTES

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

14. ABSTRACT

This thesis investigates a novel approach to identifying discriminating features of communications involving abusive hosts. The technique uses per-packet TCP header and timing features to identify congestion, flow-control, and other low-level network and system characteristics. These characteristics are inherent to the poorly connected, under-provisioned, low-end, and overloaded hosts or links typical of abusive infrastructure making them difficult for an adversary to manipulate. Supervised classifiers use these features to infer likely abusive network hosts. Prior work investigates such features to opportunistically identify inbound abusive traffic, this thesis seeks to perform active probing to generally characterize abusive infrastructure. Our approach is IP address and content agnostic, and therefore privacy-preserving to permit wider deployment than previously possible. On real-world traces obtained from accessing approximately 40,000 Alexa and 30,000 known-abusive web sites, we achieve a classification accuracy of 94 percent with a 3 percent false positive rate using only transport features. Our results suggest that transport traffic analysis can block and identify, in real-time, abusive hosts unknown to blocklists, and provide a difficult-to-subvert addition to existing schemes.

15. SUBJECT TERMS

Network Security, Supervised Learning, Abusive Network Behavior

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			19b. TELEPHONE NUMBER *(include area code)*
Unclassified	Unclassified	Unclassified	UU	93	

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

TRANSPORT TRAFFIC ANALYSIS FOR ABUSIVE INFRASTRUCTURE CHARACTERIZATION

Le E. Nolan
Captain, United States Marine Corps
B.S., Georgia Institute of Technology, 2001

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
September 2012

Author: Le E. Nolan

Approved by: Robert Beverly
Thesis Advisor

Joel D. Young
Second Reader

Peter J. Denning
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis investigates a novel approach to identifying discriminating features of communications involving abusive hosts. The technique uses per-packet TCP header and timing features to identify congestion, flow-control, and other low-level network and system characteristics. These characteristics are inherent to the poorly connected, under-provisioned, low-end, and overloaded hosts or links typical of abusive infrastructure making them difficult for an adversary to manipulate. Supervised classifiers use these features to infer likely abusive network hosts. Prior work investigates such features to opportunistically identify inbound abusive traffic, this thesis seeks to perform active probing to generally characterize abusive infrastructure. Our approach is IP address and content agnostic, and therefore privacy-preserving to permit wider deployment than previously possible. On real-world traces obtained from accessing approximately 40,000 Alexa and 30,000 known-abusive web sites, we achieve a classification accuracy of 94 percent with a 3 percent false positive rate using only transport features. Our results suggest that transport traffic analysis can block and identify, in real-time, abusive hosts unknown to blocklists, and provide a difficult-to-subvert addition to existing schemes.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

List of Acronyms and Abbreviations

3WHS	Three-Way Hand Shake
ACK	Acknowledgment Packet in TCP
ADSL	Asymmetric Digital Subscriber Line
BGP	Border Gateway Protocol
BSD	Berkeley Software Distribution
CDN	Content Distribution Network
DNS	Domain Name Service
DoD	Department of Defense
DOS	Denial-of-Service
DSL	Digital Subscriber Line
FN	False Negative
FP	False Positive
HTML	Hypertext Mark-Up Language
HTTP	Hypertext Transfer Protocol
IDS	Intrusion Detection System
IP	Internet Protocol
IRC	Internet Relay Chat
MTA	Message Transfer Agent
NPS	Naval Postgraduate School
NPV	Negative Predictive Value
P2P	Peer-2-Peer
PPV	Positive Predictive Value
PTR	Pointer Record for DNS
RTT	Round Trip Time
SVM	Support Vector Machine
SYN	Synchronization Packet in TCP
TCP	Transport Control Protocol
TLS	Transport Layer Security
TN	True Negative
TP	True Positive
TTL	Time To Live
URL	Uniform Resource Locator
USG	United States Government
VP	Vantage Point

THIS PAGE INTENTIONALLY LEFT BLANK

Acknowledgements

First and foremost, I would like to thank my loving and supportive wife Katie whose patience and understanding through these past two years has been above and beyond anything I deserve. Thank you for being there and providing me with the never ending support I needed to complete my education at NPS.

I would like to express my deep and sincere gratitude to my advisor, Professor Robert Beverly. Your guidance, mentorship, and patience during the past year has been instrumental for the completion of this thesis. In addition, I would also like to thank Professor Joel Young who spent endless hours reading and editing my thesis. I appreciate the sacrifice of so many of your red pens to the pursuit of a better thesis.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Introduction

Abusive traffic runs rampant on the Internet, in the form of email, malware, vulnerability scanners, worms, denial-of-service, drive-by-downloads, scam hosting, CAPTCHA solvers, and a myriad of other exploits. This unwanted traffic is enabled by a wide array of infrastructure including, but not limited to, complicit service providers, free web hosting providers, and botnets. Complicit service providers are those providers, generally in poorly connected countries with limited oversight, willing to host scammers, spammers, or to overlook malicious activity in. Free web providers allow for quick and inexpensive setup of a large number of scam websites. Even if some of the sites are blacklisted or shutdown to reconstituting new scam sites is trivial. Finally, botnets are used to coordinate efforts in attacks, to host or proxy content from a large number of machines, or to send out large volumes of spam.

Contemporary techniques for obtaining and using such infrastructure for abusive purposes are becoming increasingly sophisticated and economically driven. For instance, spammers receive monetary compensation for driving traffic to sites that sell knock-off jewelry, pornography, or pharmaceuticals [1]. Their motivation to maximize profit equates to the number of people they can direct to these types of scam sites. To enable their spread of spam they rely on mechanisms like botnets.

A large body of prior work, discussed in detail in Chapter 2, explores detecting abusive network behavior by, e.g., monitoring traffic content or communication patterns, while a variety of methods attempt to mitigate such traffic by, for example, distributing signature databases or tracking IP reputation. Yet, despite years of commercial and research efforts, abusive traffic continues to impart both direct and indirect damage on users, service providers, and the Internet.

1.1 Scope and Contribution

Our research investigates a new passive traffic analysis technique methodology for detecting abusive infrastructure that *does not rely on content inspection, sender reputation*, or *communication patterns* – features that are brittle and readily evaded. Instead, we use *transport-layer* traffic analysis, a technique that has shown promise in fighting abusive email [2–4].

In contrast to previous efforts [5] that leveraged aggregate traffic flow information such as that

embedded in Netflow [6] records, we find that fine-grained properties of the transport-layer packet stream, such as TCP retransmissions, packet reordering, arrival jitter, congestion and flow-control behavior, are *fundamental* sources of discriminative information that reliably characterize abusive infrastructure.

Our key insight is two-fold. First, attackers have a basic requirement to source large amounts of data, be it denial-of-service, scam-hosting, spam, or other malicious traffic. Second, the infrastructure used to support abusive campaigns must be economically viable and willing to support that traffic. As such, attack infrastructure is often comprised of botnets or complicit service providers – infrastructure with distinguishable connectivity (e.g., low and asymmetric bandwidths, congestion) and host (e.g., older computers and operating systems, high utilization) characteristics. This basic weakness manifests in the resulting traffic stream. This thesis investigates whether these fine-grained characteristics can be used to discriminate scam infrastructure from legitimate infrastructure.

Further, we contend that transport characteristics form a difficult-to-subvert discriminator as the features are related to protocols and mechanisms outside the control of the adversary. Crucially, this discriminative information is available at any point along the traffic path, including not only the network ingress, but also at the receiver and in the network core.

Because our novel method relies on previously ignored and logically orthogonal features of network traffic, it might be usefully combined with existing classification and detection mechanisms to boost performance or reduce load. For instance, the transport traffic technique can be employed as an early test before performing more costly deep packet inspection. Experimentation with combining our technique with existing mechanisms is left to future work.

In addition to providing a new method to detect abusive traffic, our approach imparts practical benefits. In particular, transport traffic analysis is content and IP reputation agnostic – and is therefore privacy-preserving – permitting use in countries with strong privacy-laws, or within the network core. For example, Directive 2006/24/EC or commonly called "The Data Retention Directive" [7] is a European Union Directive that outlines how data generated or processed using electronic communications will be stored and accessed by security agencies. The directive limits data from internet access to allocated IP address, user ID(s), name and address of the person to whom the IP was allocated, and the date/time of login of the person of interest. The content of the transmission and any identification of the destination of the communication can not be retained [8].

1.2 Goals

This thesis explores using transport network traffic features to detect abusive infrastructure. Toward this objective, we explore the following tasks and goals:

- Determine if how our approach using "fine-grained" characteristics from data calculated over the TCP header data and packet timing is superior to the "course-grained" features found in Netflow in the detection of abusive infrastructure.
- Develop a system to fetch data from a web server, while following URL redirection, and catalog the traffic behavior.
- Discuss the difficulties and limitations in determining or following redirection.
- Re-examine prior work on detecting abusive infrastructure via redirection analysis, and understand why redirection alone may not be a strong predictive feature.
- Develop a classifier using standard supervised learning algorithms to predict good traffic flows from abusive traffic flows based TCP header and packet timing characteristics.
- Develop an experiment to show that our traffic analysis approach is able to detect abusive infrastructure even when non-abusive congested flows are abundant in the training dataset with our supervised learning classifier.
- Evaluate the performance of the classification system with respect to accuracy, precision, and recall.
- Discuss what future work would benefit from our approach and what work can be done to broaden our technique.

1.3 Major Results

This paper explores the power of transport-layer traffic analysis to detect and characterize scam hosting infrastructure, including botnets. Our primary contributions include:

1. A high-speed passive traffic analysis method to detect abusive infrastructure. The technique is privacy-preserving and thus may be run both at the network edge and within the core.
2. Performance analysis of transport-layer traffic features as a detection mechanism against large, real-world data sets. We demonstrate that it is possible to detect abusive end-points with a classification accuracy of up to 94%, with only a 3% false positive rate, based on the fine-grain transport features alone.
3. A detailed analysis of the robustness of our approach. We show that it remains effective in

spite of network load fluctuations due to daily network load fluctuations and the presence of traffic intensive P2P applications.

1.4 Structure

The remainder of this thesis is structured as follows:

- Chapter 1 discusses the difficulty in the detection of abusive infrastructure. It will also introduce our technique, goals, and major results.
- Chapter 2 discusses the current research and operational practices in the field of detection of abusive network infrastructure. In addition, we compare each of the current approaches to our own. We also provide an overview of the machine learning techniques used in this study.
- Chapter 3 discusses our data collection, code validation, and traffic signal analysis methodology.
- Chapter 4 discusses our results and the real world efficacy of transport traffic analysis.
- Chapter 5 contains a summary of the larger implications of our work as well as suggestion for future research.

CHAPTER 2:
Background

2.1 Introduction

We define *abusive infrastructure* as any system that actively operates in a manner to support or perform malicious activity. These malicious activities may be known or unknown to the administrator of the system. Examples include systems that are part of larger botnets used to propagate spam, host malicious content, or even act as intermediate proxies for larger campaigns.

Effectively, Identifying abusive hosting infrastructure has become a major focus for academic and private industry. Numerous research papers and experiments have been published detailing different approaches to mitigating abusive traffic. Approaches include image shingling [9], DNS analysis [5], command and control analysis [10], transport layer characterization [3], and signature generation [11]. Many have shown great promise in both experimental and real-world situations. In this chapter we discuss some of the more prominent work in this research space. In addition, we provide a foundation on machine learning techniques and evaluation criteria relating to our approach to characterization of abusive infrastructure.

2.2 Prior Work

The following sections summarize a cross-section of current research playing a significant role in the development of the experiments contained in this thesis.

2.2.1 Image Shingling

In the space of understanding scam hosting infrastructure, Anderson *et al.* found that while a large number of nodes participate in a scam, much of their functionality is devoted to obfuscating the true origin of the scam content. Anderson *et al.* first demonstrated that a large number of nodes redirect or proxy traffic to a much smaller set of hosts [9]. Their work, SpamScatter, follows URLs within spam messages to their final landing page where the page is rendered for image shingling. Image shingling works by taking screenshots of each spam site. These screenshots are subdivided into fixed sized chunks (40 x 40 pixels) and then hashed. These hashed chunks or "shingles" are then compared across all collected spam sites to find clusters of images with similar "shingles". Based on the pre-defined cluster correlation metrics, the SpamScatter method found approximately 2,000 scams on 7,000 servers.

Using the inferred behavior of the scam hosting infrastructure SpamScatter characterizes the relationship of each host by scam type, location and blacklist inclusion. Andersen *et al.* found that individual scams are generally hosted on a single machine while a large number of hosts are used to advertise more robust scam campaigns. On the other hand, individual hosts were frequently found to support more than one scam and even act as an intermediate proxy during spam campaigns. They propose that by blacklisting or filtering a single host one could block several campaigns.

We capitalize on the SpamScatter method of extracting scam URLs from known spam on a honeypot, but augment this set with known malware and phishing sites. More importantly, our use of the URLs is to capture traffic during each stage of web fetching and redirection for the purpose of identifying discriminating features – a significantly more lightweight and non-intrusive approach than image shingling.

2.2.2 Monetization of Spam Value-Chain

Levchenko *et al.* find that the large number of nodes funneling traffic to the smaller set of nodes hosting content are often in fact part of an even smaller set of scam campaigns and organizations [1]. They discovered this by successfully characterizing the end-to-end resource dependencies and analyzing their relationships. Using three months of real-time source data gathered from captive botnets, spam feeds, and spam-advertised URLs they discovered that 95% of the spam represented three main types of goods: pharmaceuticals, replica luxury goods, and counterfeit software. They then attempted to purchase over 100 items from these sites. From the purchasing process they were able to get data about merchant bank affiliation, customer service, and fulfillment processes. They found that out of 365 million URLs there were only 45 distinct affiliate programs. For all of their attempted purchases only 13 distinct banks were used. They posited that the weak point in the scam chain is in the foreign credit card processing at these banks.

Their data collection began by extracting URLs from full-message spam feeds, URL feeds, and their own botnet-harvested spam. Then they extracted DNS information from each URL's domain and used custom web crawlers to visit each URL to retrieve HTTP behavior and landing pages. They clustered the content of each landing page and labeled the type of goods sold. Using the resultant information they were able to group sites by campaigns and by affiliate programs. The researchers then attempted to purchase actual goods from each affiliate program.

While this recent work reveals additional important properties of scams, they only focused on scams that involved selling products, typically those with questionable legality. Our traffic based approach is more general, applicable to a variety of scams while Levchenko *et al.* focused on the weak spot of the spam value chain only. We believe that our system is applicable to a variety of scams, malware, phishing, and other types of abusive traffic including active attacks.

2.2.3 DNS Behavior

The Domain Name System (DNS) [12] translates internet names to addresses, and is as essential to scams as it is to legitimate web sites. Significant prior work has analyzed various properties of the DNS to infer malicious and abusive behavior.

For example, the DNS may be used in a scam to distribute load among a large set of member bots by returning results for domain queries in a round-robin fashion. Not only does this reduce load, it improves the robustness of the system by ensuring that any single bot that is reclaimed or shut off does not impact the overall goal of serving scam content.

RB-Seeker [5] focuses on the detection of bots or botnets used in scams that attempt to obfuscate themselves from detection by using proxies and redirection through multiple compromised hosts. Hu *et al.* sought to detect these redirection bots that are impervious to the botmaster's command and control channel by analyzing DNS behavior. Their experimental results showed a high accuracy of detection with less than a 0.008% false positive rate.

RB-Seeker is compromised of 3 sub-systems. The first two sub-systems are used to accumulate domains participating in redirection or proxying: One obtains domains by following links embedded in spam emails; the other uses hypothesis testing from data collected from a campus edge router. The final subsystem takes the collected domains and uses a series of DNS probes to characterize behavior of the abusive infrastructure. Using hyperplane decision functions also known as support vector machines [13], the third sub-systems finally identifies potential malicious hosts.

In our proposed system we do not look at behavioral attributes of DNS queries. Like RB-Seeker, we do attempt to find discriminatory attributes of redirection but instead of cataloging DNS queries, we look at transport traffic features for each HTTP request. In Chapter 3 we discuss the results of our redirection attempts. Unfortunately DNS has become a poor identifier as fast-flux botnets have grown in popularity. This is one of the main reasons why we believe our system would be more effective overall.

2.2.4 Command and Control Behavior

Gu *et al.* created an approach to the detection of botnets using anomaly-based detection on two typical types of command and control channel structures: IRC based and HTTP based [10]. BotSniffer was created so that prior knowledge of a bot or botnet is not required for the effectiveness of the system. Prior knowledge includes characteristics like command and control signatures or known IP addresses of malicious servers. BotSniffer uses spatial-temporal correlation techniques in their identification process.

There exists two main components to the BotSniffer system: First is the monitor engine which sits on the perimeter of the network and generates connection records of suspicious command and control protocols, detects activity response behavior, and message response behavior. Suspicious events then are sent to a correlation engine which does a group analysis using the authors' spatial-temporal methodology. Second is the spatial-temporal correlation engine which leverages how IRC and HTTP communications are commonly used in previously discovered botnets. Using this knowledge, Gu *et al.* created their correlation engine to generalize these two types of communication techniques into two algorithms which they call "Response-Crowd-Density-Check" and "Response-Crowd-Homogeneity-Check."

BotSniffer's experimental results showed a 100% success rate in finding botnets and saw only a 0.0016 false positive rate (11 out of 6,848 servers) [10]. Unfortunately while HTTP and IRC were the standard for many years, recent research has shown that peer-to-peer has become the new standard for command and control communications. This distributed type of system was outside of the scope of their paper and was cited as future work.

As noted above, Current botnets have evolved passed IRC and HTTP communications channels into peer-to-peer communications making botnets harder to identify. Unlike BotSniffer, our approach does not require knowledge of the command and control channel.

2.2.5 Signature-Based Detection

Signature-based detection is commonly used in worm and virus detection systems. Xie *et al.* extended this approach into a system called "AutoRE." [11] AutoRE focuses on characterizing spamming botnets by creating regular expression signatures derived from URLs mined in spam payloads. In addition, AutoRE groups spam emails into spam campaigns. Spam campaigns are focused on targeted spam efforts on a single product or service.

AutoRE has three modules: a URL pre-processor, a group selector, and a regular expression

generator. The URL pre-processors mines URLs from spam and then groups them based on their web domain. Group selection is used to identify specific spam campaigns. This is accomplished by greedily associating URLs together via temporal correlation. Finally, given a set of URLs pertaining to a given domain, regular expressions are created.

The regular expressions are created by constructing keyword-based signature trees, generating candidate expressions, and then evaluating the expressions to ensure specificity. Xie *et al.* found that their system successfully identified 7,721 botnet-based spam campaigns with 340,050 unique botnet host IP addresses in their 3 month sample of spam directed toward the Hotmail web-mail service. Their false positive rate was between 0.0011 and 0.0014 [11].

Unlike autoRE, our system is content agnostic and does not rely on brittle heuristics to determine malicious activity. AutoRE only looks at the URL of a domain and disregards other information available in the traffic streams. Our system on the other hand focuses on the actual behavior of the network traffic and by avoiding content analysis preserves privacy.

2.2.6 Graph Theoretic Approaches

Zhao *et al.* tackled the difficult problem of identifying bots used to signup for free web mail from major providers as part of scam campaigns [14]. By only sending a few messages from each account on well-known and trusted providers they attempt to evade detection. Their system BotGraph detects such accounts by constructing large user-user graphs and looking for tightly connected subgraph components.

BotGraph is based on the assumption that bot-users share similar IP addresses when logging in and sending emails from web based accounts. BotGraph leverages this type of sharing to detect botnets. To enable the massive computation overhead needed to mine the log data of major web email services Zhao *et al.* developed an efficient distributed system to construct and analyze large graphs. In one of their experiments they used 240 machines to analyze a 220 GB Hotmail log in 1.5 hours. In another experiment on 2 months of Hotmail logs (450 GB), BotGraph was able to identify 26 million bot-accounts with a false positive rate of 0.44% [14].

Their methodology focuses on the origination point of spam as discoverd by mining web email logs and does not attempt to infer any information from the hosting infrastructure traffic. Unlike our approach, theirs requires significant time and resources to identify suspicious bots, our approach is relatively light-weight in both time and resource costs.

2.2.7 Behavior Clustering

The approach created by Gu *et al.*, called BotMiner, is based on the analysis of the communication structure found in network traffic [15]. Combined with IDS-like functionality at higher layers, BotMiner clusters hosts with similar flows where the intuition is that bot hosts have communication patterns differing from legitimate hosts. BotMiner generalizes the essential properties of botnets in an effort to detect bots without *a priori* knowledge of the botnet itself.

BotMiner uses cross-clustering correlation from two network monitors called the "A-plane" and "C-plane." The "A-plane" is responsible for detecting suspicious activity (scanning, spam, binary downloads, and exploits) and the "C-plane" is responsible for logging network flows. Both monitors extract features from their respective data and by using clustering algorithms, they attempt to find groups of machines with similar behavior. The final step is cross-plane correlation analysis to cross-checks the two planes and find intersections representing evidence of botnet activity.

BotMiner was tested and evaluated against 8 known botnets consisting of 4 IRC-based, 2 HTTP-based, and 2 P2P-based nets. These botnets were overlayed over normal network traffic. The authors then took traffic traces of a random mixture of hosts consisting of both normal and bot based behavior. They tested BotMiner daily over a 10 day period and were able to successfully detect all 8 botnets. 6 of the 8 botnets were detected 100% of the time while the other two were detected at a rate of 99% and 75%. The overall false positive rate was 0.003%.

Unlike BotMiner, our approach for identifying abusive infrastructure is independent of the communication structure of the malicious activity. Our focus is on the properties of the traffic streams and we ignore application-layer content and IP addresses entirely. Our system attempts to minimize all privacy concerns yet is effective enough to discriminate abuse on the network.

2.2.8 Transport Layer Characteristics

Rather than relying on IP reputation or content inspection, Beverly *et al.* created a spam detection technique using discriminating features inferred from TCP headers and packet timings for each communication flow [3]. For spammers to be successful they must send large volumes of emails causing contention and congestion on a network. These effects are exaggerated for many botnet hosts which reside on residential broadband networks where there are large gateway buffers and asymmetric bandwidth.

Transport-layer properties like lost segments, round trip times, and jitter variance exhibit be-

havior that differ from normal usage by ordinary user. Using statistical analysis of a variety of transport-layer properties Beverly *et al.* demonstrated that an auto-learning classifier can distinguish spam email from normal email with a accuracy rate of over 95%.

The authors' research is the most closely related to our proposed system. Whereas this technique has been applied to the reception of abusive messages, we investigated the logical dual: whether traffic features can identify host serving abusive content. As we will demonstrate, the technique generalizes well, but hones in on substantially different features from those of spam identification.

2.3 Supervised Learning

Many abusive infrastructure detection systems rely on machine learning techniques [16]. In our research, we employ supervised learning to determine if a network TCP flow is abusive or not. Supervised learning is a task of machine learning where an inferred function or classifier is created [17]. The supervised learning (unlike unsupervised learning), model takes in labeled training data with two parts: a vector of (hopefully) discriminating values and a label. After a supervised learning algorithm (e.g. Näive Bayes) processes the training data the classifier then can predict the correct label for a new input vector with a certain degree of accuracy. The accuracy of the classifier can be tested by using statistical analysis like k-fold cross-validation [18].

The vector of discriminating values for our training data consists of features extracted from the data we mined from the TCP headers and packet timings from individual network flows. These 19 features are outlined in 3.4. The label for each vector designates the vector as either malicious or good. In the context of this research we employ Näive Bayes, and C4.5 which are well-known supervised learner algorithms. In our research we used a 10-fold cross-validation test to validate the accuracy of our technique.

2.3.1 Näive Bayes Classifier

The Näive Bayes classifier [17] is based on Bayes' theorem and the assumption that each feature is conditionally independent of every other feature, given the class variable C. Baye's theorem [19] is defined as:

$$P(C = c_k | \overrightarrow{F} = \overrightarrow{f}) = \frac{P(\overrightarrow{F} = \overrightarrow{f} | C = c_k) P(C = c_k)}{P(\overrightarrow{F} = \overrightarrow{f})},$$

where C is the class variable and c_k is the variable for each individual flow labeled as "legitimate" or "abusive". $\overrightarrow{F}$ is the feature vector and $\overrightarrow{f}$ represents an individual flow vector where $f_1, ..., f_n$ are the values of the attributes. Applying the independence assumption:

$$P(\overrightarrow{F} = \overrightarrow{f}|C = c_k) = \prod_i P(\overrightarrow{F_i} = \overrightarrow{f_i}|C = c_k),$$

and by using the maximum a-posteriori probability (MAP), the basic decision rule can be defined as follows [20]:

$$c = classify(f_1, f_2, ..., f_n)$$

$$= \underset{k=(abusive,legitimate)}{\text{argmax}} (P(C = c_k|\overrightarrow{F} = \overrightarrow{f}))$$

$$= \underset{k=(abusive,legitimate)}{\text{argmax}} \left(\frac{P(\overrightarrow{F} = \overrightarrow{f}|C = c_k)P(C = c_k)}{P(\overrightarrow{F} = \overrightarrow{f})} \right)$$

$$= \underset{k=(abusive,legitimate)}{\text{argmax}} \left(P(C = c_k) \prod_i P(\overrightarrow{F} = \overrightarrow{f}|C = c_k) \right)$$

The prior probability $P(C = c)$ is given by the ratio of the number of examples that belong in class c to the total number of examples. The product of the conditional probabilities depends on the feature types, whether they are discrete or continuous. If the features are discrete, the conditional probability is the ratio of the number of vectors F_i that have value f_i and belong to class c_k to the total number of vectors that belong to class c_k. In the case of continuous values, we assume that they follow a normal distribution and we have [17]:

$$P(\overrightarrow{F} = \overrightarrow{f}|C = c_k) = g(\chi_i; \mu_i, c_k, \sigma_i, c_k),$$

where

$$g(\chi;\mu_X,\sigma_X) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

is the normal (Gaussian) distribution.

2.3.2 C4.5 Decision Trees

Decision trees are used in machine learning as a classification model that attempts to predict an output value of a target item based on a set of observations or input variables. A specific type of decision tree which is commonly used in supervised learning is called C4.5. C4.5 [21] creates a decision tree classifier using what is called information entropy. C4.5 was developed by Ross Quinlan and is an extension of his earlier ID3 algorithm.

The algorithm evaluates each feature vector based on the following strategy: initially, it selects the best feature as the root of the decision tree. For every different value of the feature, it creates a descendant node, which consists of all the vectors that contain the specific feature value. This whole process is repeated recursively for each feature node in the decision tree. The process ends when one of the following conditions is met:

1. all vectors of the current node belong to the same class or
2. all features are used

How well the decision tree performs depends on the selection process of the best feature. A suitable measure for the evaluation of the features, and therefore for the selection of the best feature, is the information gain (IG) of an attribute A [17]. If we define S as the set of training examples, then the mathematical representation of IG is given by the following formula:

$$IG(A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v),$$

$$Entropy(S) = -\sum_{i=1}^{c} p_i \log_2 p_i,$$

Values(A) is the set of attribute A values, S_v is a subset of S that contains the examples with attribute A having value v, and p_i is the ratio of the number of examples that belong to class i to the total number of examples. Entropy represents the amount of information that is provided

by the attribute and, in information theory, is measured in bits [17]. Our goal is to maximize the IG of the selected attribute by minimizing the entropy of S_v, or, in other words, by reducing the number of bits. Information gain, however, has the disadvantage that it selects attributes with a large set of values. To overcome this shortcoming, Quinlan [22] suggests utilizing the information-gain ratio, which is formalized as follows:

$$GR(A) = \frac{IG(A)}{IV(A)}, \text{where}$$

$$IV(A) = -\sum_{i=1}^{|A|} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|},$$

and S_i are the subsets of S that contain attribute A with value i. So, again, our goal is to find the attribute that maximizes the above ratio.

2.4 Evaluation Criteria

Once the classifiers are trained, we must have a way to evaluate their performance and compare those of different experiments. Standard performance metrics include precision, recall, F-score, and accuracy.

Accuracy is the rate of correct predictions made by the classifier over a data set. It is obtained by dividing the number of correctly classified flows by the total number of flows in the set.

Precision measures the proportion of items correctly classified as belonging to a particular class, i.e. the number of items correctly labeled as a class divided by the total items labeled as that class.

Recall measures the proportion of items belonging to a particular class that the classifier actually identified, i.e. the number of items correctly labeled as a class divided by the total number of those items in the data set. The formulas for precision and recall follow:

$$precision = \frac{TP}{TP+FP}$$

$$recall = \frac{TP}{TP + FN}$$

Where TP is true positive, the number of items correctly assigned to a class. FP is false positive, the number of items incorrectly assigned to a class. FN is false negative, the number of items of one class identified as a member of another class.

F-score combines these two measures into one evenly weighted metric. This prevents the experimenter from making design adjustments that favor one measure or another. F-score is the harmonic mean of the precision and recall:

$$F = 2\left(\frac{precision \times recall}{precision + recall}\right)$$

2.5 Conclusion

In this chapter we summarized a few of the prominent research approaches to the detection of abusive network behavior. We have also provided background information on supervised learning and two classification algorithms (Näive Bayes and C4.5 Decision Trees) that were used during our research. In addition, we discussed the foundation of how we evaluated our classifier. In the next chapter we will discuss in detail our experimental methodology to the detection of abusive infrastructure.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 3:
Experimental Methodology

This chapter describes our experimental methodology designed to test our hypothesis that we can use fine-grained properties of the transport-layer packet stream in conjunction with supervised learning-based classification to find abusive network infrastructure. We begin by describing how we obtained our host population dataset. Next we show how we captured our traffic flows from this dataset from multiple vantage points. Then we discuss our transport traffic classification procedure focusing on how we build our supervised- based classification model. Finally we discuss how we evaluated our technique using statistical analysis.

3.1 Host Populations

To use a supervised learning-based classifier, we need to collect a set of training data where the label of the input feature vector is known to be either abusive or non-abusive. Our features are the set of TCP and packet timing characteristics mined from HTTP flows of known domains. To obtain this dataset we collected information from the following sources:

1. Spam Honeypot
2. External Known Malicious Domains and Phishing Sites [23] [24] [25]
3. Alexa [26]

Our first population of domains were mined from a Spam honeypot that we managed from the Naval Postgraduate School. This honeypot did not have any externally registered valid email accounts, therefore any emails received by the honeypot are considered abusive in nature. Each received message is processed to extract all embedded URLs. To disguise itself as legitimate traffic, many emails embed images and URLs from legitimate websites. In addition, to minimize bandwidth costs by sending massive amounts of emails spammers use hypertext references to external images and style sheets. To ensure that we capture only URLs of abusive hosts we check a whitelist before flagging a URL as abusive. The whitelist is a manually evolving list of URLs deemed to be legitimate but found in spam messages. Once the URL is determined to be abusive its information is saved into a database (MySQL) for further experimentation.

Abusive hosts encompass more than just spamming and scamming infrastructure. We also test on domains from two well-known malicious domain sources: `phishtank.com` [25] and

malwaredomains.com [23]. phishtank holds a repository of current malicious phishing domains. Phishing is fraudulent attempting to get one to provide personal information, including but not limited to, account information. This is different from regular spam which generally are unsolicited commercial email. malwaredomains maintains a listing of domains known to be used to propagate malware and spyware. Both phishtank and mawlaredomains update their repository information on an hourly and daily basis respectively. Before deploying domains and hosts to the data collection portion of the experiment, we ensure that we have the most up-to-date set of information to decrease the possibility that domains are no longer active. In Section 3.2 we discuss in more detail the actual number of domains that were still active during our experiments.

From the spam honeypot we were able to obtain over 1.4 million URLs over a period of a few weeks. We compacted this number down to 9,679 fully unique URLs by removing duplicate domains and whitelisted URLs. The whitelists were used because in many cases spam incorporates legitimate URLS. Next, combining the phishtank and malwaredomains URLs resulted in a total of 20,013 unique abusive URLs. We found that many of the URLs occurred in both lists and removed any duplicates as necessary. The spam list and the combined phishtank/malware list (which we title as the "external" list) make up our abusive host "ground truth" domains.

For the legitimate hosts we use a subset from the listing of most visited websites reported by Alexa, subsidiary company of Amazon.com that maintains analytics of web traffic. Alexa's estimates and ranking system are based on internet users using Alexa's downloadable toolbar application (reported to be in the millions of users) [26]. Using the number of unique users who visit a site per day and the total number of user requests for a specific site gives Alexa a realistic sample of all internet users.

We created two separate lists from the Alexa domains. The first list is the top 20,000 most popular websites. The second list is made from a random sampling of 20,000 hosts from their top one million websites. Random sampling yields a list of legitimate hosts residing on infrastructure of varying capabilities rather than the top sites that are generally run from major corporations with large hosting capabilities. The random selection gives a robust view of the types of websites users comes into contact with on a daily basis.

3.2 Data Collection

For each host in the spam honeypot, malware lists, and Alexa lists, we capture all packets in the network flows generated by visiting that host. We implement a custom URL content fetcher and redirection follower. This custom fetcher program was created in the uses standard HTTP request protocols emulating a conventional web browser. The program disguises its user-agent as "`Mozilla/5.0 (Windows NT 6.1; WOW64)`" in case any web server or proxy is filtering requests that it perceives as non-human initiated such as a web crawler. As the fetcher initiates each connection we record network flow information using `tcpdump` and save the information in a standard PCAP file. In addition, we save identifying flow information (e.g. IP,Port,Redirection Parent) in a MySQL database for tracking of redirection chains for further analysis. Furthermore, we ignore all robot exclusion tags.

Figure 3.1 illustrates precisely how our data collection process works. We begin by using our customized URL content fetcher and follower. We follow each URL in our domain corpus to its logical end which includes following any redirection. We traverse the following types of redirection:

1. 3XX HTTP "content moved" status codes
2. HTML meta-refresh
3. HTML frames
4. Non-obfuscated JavaScript

The fetcher respects cookies and supports Transport Layer Security (TLS) when required, allowing us to successfully follow redirection chains where abusive hosts attempt to employ these for filtering. For example, we found a few cases where access to a final landing web page in a URL redirection chain requires a cookie to be set from the previous page in the chain. If a user were to directly access the final landing page they would be either given an error status code, or redirected somewhere else. Our JavaScript redirection covers traditional calls to:

- `window.location`
- `location.href`
- `location.replace`
- `window.navigation`
- `self.location`
- `top.location`

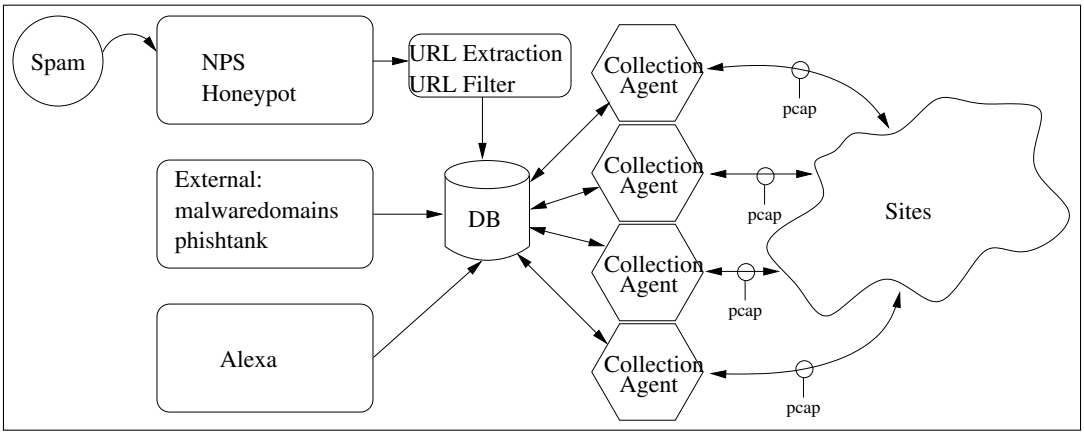


Figure 3.1: Data Collection: Multiple sources of abusive and legitimate URLs are probed by collection agents (Table 3.1) with varied connectivity. Contemporaneously, collection agents capture packets.

Where JavaScript was intentionally obfuscated, we do not follow any redirection. We discuss the prevalence of obfuscation in Section 3.3.

From January 25 to 27, 2012, we deployed our fetcher on four separate collection agents, running FreeBSD and Linux TCP stacks, termed "vantage points." Each "vantage point" independently fetched each known URL from our collected domain list. As detailed in Table 3.1, each agent has different network connectivity and physical attachment points. In addition to examining detailed characteristics of the network traffic stream, including congestion and delay effects, the collection agents are also span three orders of bandwidth magnitude, different multiplexing technologies, located on both east and west coasts of the United States, and are connected via academic, commercial, and residential ISPs. We examine the relative performance of each collection agent and quantify our system's sensitivity to legitimate sources of congestion in Section 4.4.

During the actual collection process the network packets are saved into bidirectional flows according to distinct tuples of:

- destination IP address
- ephemeral source TCP port

Our flow definition differs slightly from common usage (e.g., Netflow) as our collection agent's IP addresses and remote TCP port are fixed. As flows complete, either via an explicit TCP

Table 3.1: Collection agents: a variety of vantage points and connectivity types that impact traffic classification.

ID	Connectivity	Location
VP1	1.5Mbps Commercial	CA, USA
VP2	University 1Gbps	MA, USA
VP3	12/20Mbps Residential	CA, USA
VP4	University 1Gbps	MA, USA

termination or via a 300 second timeout, we extract transport-layer features. The features that are extracted are listed in Table 3.6. By recording the ephemeral port during the content fetch, we match flows to their respective content.

We found that many of the domains extracted from the spam honeypot and external abusive domains found in the "external" list proved to be relatively unreliable; Approximately 50–70% of the URLs that we attempted to fetch were successful. The results are summarized in the "yield" row in Table 3.3. The final row in Table 3.3 shows the total number of fetches per population when including redirection chains.

3.3 Fetcher Validation

To validate that our custom fetcher is following URLs and capturing content correctly we record all exceptions the program encounters. The most common types of exceptions we found were involved in the initial three way handshake between the vantage point and the hosting server and include the following:

- timeouts
- DNS failures
- connection resets
- connections refused
- route failures
- redirection errors

Table 3.2 is a list of the percentage of the total errors that occurred during the initial connection.

Most exceptions we encounter are the result of connection failure. Table 3.4 breaks down the percentage of failed fetches per population by initial URL fetch in a redirection chain and total fetches of all URLs. For example, out of all of the attempted fetches from the Honeypot

Table 3.2: Total Errors During Initial Connection

Dataset	**Error Percentage**
Alexa	90%
Alexa Random	94%
Honeypot	93%
External	99%

Table 3.3: URL populations for data collection

	Alexa		**Abusive**	
	Top	**Random**	**Honeypot**	**External**
Count	20K	20K	1.4M	20K
Unique	20K	20K	9.7K	20K
Yield	19.3K	19.1K	6.5K	9.4K
Total Fetches	27.9K	25.3K	7.9K	12.5K

population, only including the initial URL in redirection chain, 31% failed and 3% of the total failures were non-connection related. For total URLs fetched, 30% were failures and 3% were non-connection related. This results show that approximately 90% of the exceptions from the Honeypot are connection based.

Examples of non-connect exceptions include:

- bad HTTP status codes
- missing or blank redirection locations
- malformed redirects

Table 3.4 further divides the frequency of connection exceptions among our URL populations for both the initial URL, as well as the initial and all URLs along the redirection chain. We omit any fetches that result in an exception from our analysis. Surprisingly, we observe 751 exceptions among the Alexa top 20,000 sites. A full list of the exception counts for each dataset are found in Table 3.5.

The random selection of 20,000 Alexa hosts reveals roughly double the exception error rate, an unsurprising result as we expect more popular sites to be hosted on more reliable infrastructure. The exception rate among the honeypot and external URL populations is much higher, as is expected for abusive, unreliable infrastructure. Most importantly, the frequency of exceptions stemming from causes other than failed connection is under 1% for all populations except the

Table 3.4: Fetch exceptions per population, divided by initial URL (Init) versus all URLs followed (Init+Redir). The majority of exceptions are connection related.

Name	**All (Init)**	**All (Init+ Redir)**	**Non-Connect (Init)**	**Non-Connect (Init+Redir)**
Alexa	4%	3%	1%	1%
Alexa Random	8%	7%	1%	1%
Honeypot	31%	30%	3%	3%
External	53%	46%	1%	1%

honeypot URLs where the failed connection rate is under 3%. These results reinforce our confidence that the customized fetcher is working as designed.

Table 3.5: Exceptions by Type per Dataset

Exception	**Alexa**	**Alexa Random**	**Honeypot**	**External**
Timeouts	357	479	855	1,680
DNS Failures	327	423	2,279	8,674
Connection Resets	16	13	0	97
Connection Refused	20	29	47	176
Route Failures	11	1	4	21
Redirection Errors	20	13	69	28
Total	751	958	3,254	10,676

To further validate our data collection system we perform manual, random sampling. We take a sample of 250 URLs from each of the four datasets in Table 3.3 (1000 random samples in total) and perform a side-by-side comparison between our fetching engine and a default Firefox 10.0 web browser with JavaScript enabled. We analyze URLs and content of the final page in the redirection chain for both the fetcher engine and the web browser. In all cases, we accurately capture instances where no redirection occurred and in cases where redirection occurred due to HTTP status codes or HTTP header location changes.

The one area we did find inconsistencies was when JavaScript was used for redirection. After noticing inconsistencies we randomly sampled another 350 domains from the Honeypot/Malware list but this time looking at domains only flagged as having JavaScript redirection. We found that 20 (6%) of the JavaScript sites inspected were obfuscated using techniques such as hex value

replacement and text splitting by using variable manipulation. In general, however, JavaScript in abusive sites was either, clearly redirection, or used for other functions. In Alexa sites we manually sampled, we found no obfuscation for sites using JavaScript for redirection.

3.4 Transport-Level Signal Analysis

The intuition behind our transport traffic analysis technique is simple. Operators of abusive infrastructure and the perpetrators of various scams are faced with economic and policy constraints: they must find hosts that will support their abusive traffic at a cost that allows them to be profitable, or to otherwise achieve their objectives. The resulting abusive infrastructure is frequently either compromised hosts, for instance, rent-per-hour botnets, or complicit service providers. In either case, these hosts are typically poorly connected, under-provisioned, and overloaded (in terms of either communication, computation, or both).

For example, botnet hosts can be unwittingly compromised home users' computers. That is residential machines without enterprise-level infrastructure. These hosts are resource constrained, both in terms of available computational power and bandwidth. Moreover, these residential internet connections typically have asymmetric bandwidth, e.g., aDSL or cable modem-links that convey distinct signatures to their traffic stream. The major reason for the asynchronous bandwidth is generally due to user requirements. The common usage for the internet is to download information like email, news, music, and video. The bytes required for the request is tiny relative to the bytes to be downloaded. Similarly, complicit service providers willing to support abusive infrastructure are frequently located in countries and at companies with poor internet connectivity, low bandwidth, and with limited peering.

The fact that botnets must send large volumes of traffic, whether spam messages, dictionary attacks, vulnerability scans, scam website hosting, or denial-of-service attack packets exacerbates the problem of poor connectivity. As malicious traffic congests available capacity, local queues form in the host operating system and residential gateway. In fact, the local buffer sizes on residential cable and DSL modems is quite large. This phenomenon is called "bufferbloat" [27]. It is reasonable to expect:

- TCP timeouts
- retransmissions
- resets
- out-of-order packets

- highly variable round trip times (RTT) estimates

This traffic signature is distinguishable using statistical learning techniques. Table 3.6 details the complete set of features we collect on a per-flow basis. Some of the features are bidirectional, for instance packets and bytes, while others only have meaning relative to the remote host, e.g., receiver window. The last three features, TTL value, IP "don't fragment," and SYN/ACK size correspond to features typically used for passive host operating system inference [28]. While we collect these values, we omit them from any analysis in this paper as these are *easily mutable* and less fundamental than the other features which are more difficult for abusive hosts to subvert.

Likewise, we do not make use of reputation measures such as IP addresses, or perform inference over IP addresses, which are known to be unreliable [29]. Further, our technique does not require deep-packet inspection, a costly measure with privacy implications. Instead, using a transport-level approach imparts several important benefits:

1. By analyzing the transport traffic stream, our technique is privacy-preserving and therefore may be run in the network core rather than at the edge. Privacy-preservation is crucial to many environments, e.g., satellite networks, and in many political settings, e.g., current European privacy laws.
2. The ability to run in the network core has the potential to stanch malicious traffic before it saturates access links.
3. This technique can serve as a fast discriminator for in-core use, or as a distinct signal for boosting edge classification performance.
4. By exploiting the root-cause of the attacks, the traffic itself, operators of abusive infrastructure must either acquire more nodes or send traffic more slowly; either of which imposes cost.

3.5 Prediction

Our extracted transport traffic features are used as the input to traditional supervised, train-then-test, learning algorithms. We describe the various methods to form predictions over the traffic features here.

Classification and Feature Selection

The result of our feature extraction process is a per-flow (f_i) feature vector $\mathbf{x}_i$ containing each of the values described in Table 3.6. We use two well-known statistical learning algorithms: Naïve Bayes, and C4.5 Decision Trees. These two algorithms are discussed in detail in Section 2.3.1

Table 3.6: Transport Traffic Features

Feature	BiDir	Description
Pkts	Y	Packets
Bytes	Y	Bytes
Rxmits	Y	Retransmissions
RSTs	Y	Packets with RST bit set
FINs	Y	Packets with FIN bit set
RwndInit	N	Initial receiver window
RwndMin	N	Minimum receiver window
RwndAvg	N	Average receiver window
Rwnd0	N	Times zero window advertised
MaxIdle	N	Max idle time between pkts
3WHS	N	Initial round trip time estimate
Dur	N	Total flow duration
RTTVar	N	Variance of per-segment RTT
JitterVar	N	Variance of inter-packet delay
TTL	N	IP Time-to-live (fingerprint)
DF	N	IP Don't fragment (fingerprint)
SAsize	N	TCP SYN/ACK pkt size, w/opts

and Section 2.3.2. We use these algorithms as implemented in a third party software suite, Orange [30]. Orange is an open source data mining software suite that with a native Python language interface.

To validate results from our classifier, we employ 10-fold cross-validation in all experiments. The randomized flow features are partitioned into ten sets. Nine of the ten sets serve as training to build the learned model, while testing is performed on the held-out fold for validation. The prediction accuracy from each fold combination is averaged to provide a final performance number. In this way, we ensure that our results generalize without respect to a particular training and testing split or the composition of a particular training set.

Often, one wishes to understand which feature is most discriminative power that is which properties of the traffic stream are predictive. Some learning algorithms, such as decision trees, naturally provide the best features as part of their output. In our experiments we make use of the Relief algorithm [31] for best feature selection. This is the standard algorithm feature selection algorithm in the Orange data mining software [30].

Performance Metrics

We use standard classification performance metrics. We call an abusive host "positive" and a legitimate host "negative" to indicate disposition. Correct predictions result in either a true positive (tp) or true negative (tn). An abusive host that is mislabeled as legitimate produces a false negative (fn), while a legitimate host misclassified as abusive produces a false positive (fp). Note that false positives are particularly onerous as there is a high cost to blocking or filtering legitimate sites. As performance metrics, we consider accuracy (Acc), precision (Pre), recall (Rec), false positive rate (FPR), and F-score:

$$Acc = \frac{tp+tn}{tp+fp+tn+fn} \tag{3.1}$$

$$Pre = \frac{tp}{tp+fp} \tag{3.2}$$

$$Rec = \frac{tp}{tp+fn} \tag{3.3}$$

$$FPR = \frac{fp}{fp+tn} \tag{3.4}$$

$$F-score = 2\left(\frac{Pre \times Rec}{Pre+Rec}\right) \tag{3.5}$$

Conclusion

In this chapter we discussed our experimental methodology. We began with our approach to the collection of both abusive and non-abusive hosts. The next step of our methodology is the generation of transport flow data from accessing the collected host set using our custom URL content fetcher and redirection follower. From the transport flow data we extract 17 features that we use as the input vector into our supervised learning classifier. The classifier is then tested using a 10-fold cross validation technique that is common in statistical analysis. In addition, we discussed the validation of our fetching software and how we determined the performance of the classifier.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 4:
Results

This chapter examines our experimental results. We begin with a discussion of our approach compared to those used with Netflow records, and in particular those relying on redirection behavior evident in Netflow. Next, we examine the types and frequency of redirection found in our datasets. Then we show how our classification is applied to HTTP traffic gathered from both known good and abusive hosts from four topologically separated vantage points. In addition to classifier performance, we discuss which features were most discriminative, and the sensitivity of our model to legitimate sources of congestion.

4.1 Netflow Limitations

As discusses in Chapter 2, significant prior work has explored various aspects of network-level information to infer various malicious activity. We stress here our approach is distinct from current practice: rather than examining the distribution of abusive IP addresses [32], network BGP prefixes [29], IRC channel [10] or peer-to-peer botnet signatures [15], DNS behavior [33], or web page image shingling [9], our approach is content, name, and IP address agnostic. By not having to rely on private content, our system can bypass issues which generally hinder other approaches including legal constraints or data encryption.

Most closely related to our effort is work using Netflow [6] information to infer suspicious behavior. Netflow records are created per unique flow tuple consisting of: source and destination IP address, protocol, and source and destination transport port. Each directional flow record contains coarse-grained information including the total number of bytes, packets, and the flow duration. Netflow records contains only a small subset of the features we collect and analyze in this research (listed in Table 3.6). Netflow version 5 is one of the most common versions used today; its features are shown in Table 4.1.

One example of using Netflow records for abusive botnet host detection comes from "redirection bot seeker," [5] or RBseeker. In RBseeker, Netflow records are used to identify infrastructure bots using redirection, as described in Section 3.2, to obfuscate and hide the true source of abusive content. Because many schemes employ Netflow, it is our intent to show our fine-grained packet feature approach is superior to Netflow analysis in redirection. We will demonstrate this superiority by comparing our technique to RBseeker.

Table 4.1: Pertinent Fields in Netflow v5 Record

Feature	Description
Source IP	IP address of the device that sent the flow
Destination IP	IP address of the destination device
Next Hop IP	IP address of next device
Inbound snmpIFindex	SNMP index that identifies the Inbound Interface
Outbound snmpIFindex	SNMP index that identifies the Outbound Interface
Packet Count	Number of Packets in Flow
Byte Count	Total Number of Bytes in Flow
Time as Start	Value of SysUpTime when first packet received
Time at End	Value of SysUpTime when last packet received
Source Port	Port number of the device the flow went out of
Dst Port	Port number of the device the flow went out to
TCP flags	Protocol State
Layer 4 Protocol	Type of layer 4 protocol
TOS / Diffserv	Value that designates special handling
Source AS	AS that flow came from
Destination AS	AS that flow is going to

4.1.1 Per-device Redirection is Common

An investigation of the top 20,000 Alexa sites suggests redirection is more common for legitimate web traffic than initially thought. One root cause of legitimate redirection is websites using redirection as a mechanism to tailor content to the end-user's browser and device. For example, a user on a mobile device accessing the website of a newspaper will commonly be redirected to a mobile version of the content suitable to the device's screen, network speed, or user preference. User preference can be dictated by their service plan (e.g., different speeds or content views based on how much their service subscription costs). Over the past few years, content for legitimate websites have been become more dynamic and one of the tools content distributors employ is redirection to satisfy not only customer desires, and company profits, but also infrastructure capability (e.g., mobile devices versus desktop computers).

To understand redirection due to end-user device type, we created a HTTP poller which present 12 different user-agent strings to each of the top Alexa listed web servers. As part of the exchanged HTTP headers, the user-agent header field advertises the software and hardware of the requesting browser [34]. We test user-agent strings representing all major browsers, operating systems, and a variety of mobile devices.

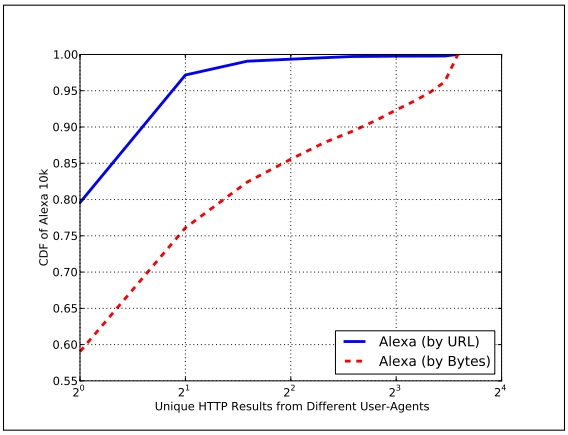


Figure 4.1: Redirection by per-device customization.

For each web site, we count the total number of unique results returned in terms of redirection URLs and bytes returned, i.e., we measure the size of the returned set. Note, we are not measuring if there is *any* redirection, but rather if the final landing page is *different* for different devices. Thus, if the server's behavior is consistent (e.g., always redirecting) irrespective of the user-agent, the set size is one. In contrast, a site that for example redirects users when they report to be using a mobile device increases the set size.

Figure 4.1 demonstrates that approximately 20% of Alexa listed sites employ at least one user-agent specific redirection. Similarly, more than 40% of the sites return different byte counts depending on the advertised browser. Device or web browser type *does* play a significant role in observed redirection, and will impact schemes which rely on identifying such redirection among traffic flows.

4.1.2 General Redirection is Common

User-agent redirection tells us using redirection as a sole indicator of abusive content is not completely accurate. In addition, we wish to understand the types and prevalence of various redirections encountered when visiting both legitimate and known abusive web sites in Table 3.3. Note, in this part of the analysis, our fetcher presents only a single HTTP user-agent string to each web server as discussed in Section 3.2. In other words, we next seek to ascertain redirection not due to user agent.

Figure 4.2 shows the fraction of sites which perform any type of redirection for our various datasets, as a function of vantage point. The results show that non-abusive hosts redirected ap-

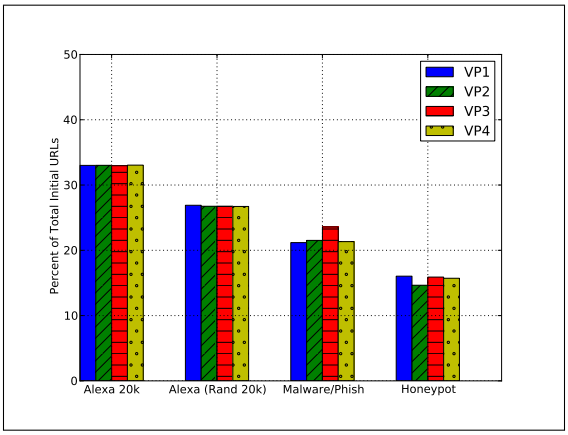


Figure 4.2: Frequency of redirection among web site populations, as observed from four vantage points.

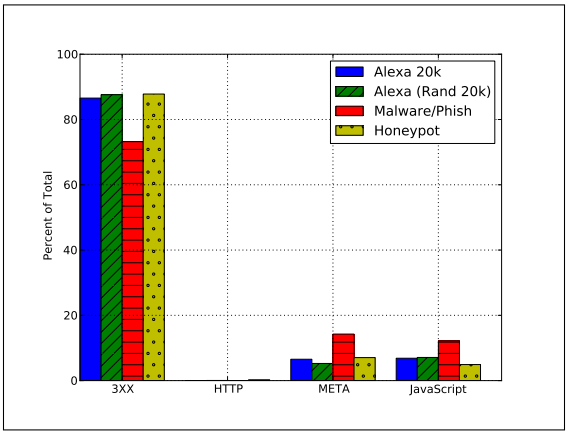


Figure 4.3: Redirection technique employed when a web site redirects.

proximately 32% (Alexa Top 20K) and 27% (Alexa Random 20K) of the time, while our abusive hosts redirected approximately 20% (Malware/Phishing) and 15% (Honeypot) respectively.

Based on vantage point we see the non-abusive hosts showed almost no difference in redirection behavior while the abusive sites do show slight variability. The variability for the Malware/Phishing and Honeypot sites are attributable to site availability or site change due to fast-flux DNS changes. In a small number of instances we found URLs would resolve at one vantage-point but not from another. Also in certain cases, a URL which resolved in one vantage-point had a different IP address at another vantage-point.

Table 4.2: HTTP 3xx Redirection Status Codes

Feature	Description
300	Multiple Choices - user select preferred representation
301	Moved Permanently
302	Found
303	Response Can be found under different URI
304	Not Modified
305	Use Proxy
306	Unused
307	Temporary Redirect

Taking these examples into account, we believe site availability and fast-flux DNS are relevant to abusive infrastructure detection.

Before experimenting, we posited that redirection chain statistics play a role in discrimination. Surprising, however, we observe the Alexa top 20K sites perform the most redirection, while the malware and honeypot sites redirected approximately 12% less. If redirection was once a sufficient means to detect abusive infrastructure, abusive sites have compensated and now use different obfuscation techniques, e.g., proxying and DNS redirection.

Note the Alexa database includes domains rather than specific web sites. For this analysis, we prepend "`www`" to each domain before probing. Without such prepending, we observe significantly higher redirection rates, approximately 70% for Alexa listed sites. Thus, a third source of redirection is the common practice of users to enter the abbreviated, domain-only form of a URL, only to be redirected to a "`www`" URL.

Another aspect of redirection we investigated is the distribution of types of redirection techniques irrespective of legitimate or abusive site. The four techniques we focused on are discussed in Section 3.2. Overwhelmingly, we found the technique for redirection was via HTTP `3xx` status codes. The `3xx` status codes are listed in Table 4.2. As shown in Figure 4.3, approximately 87% of the redirecting Alexa sites uses status codes, as compared to 73% of malware and 88% of honeypot sites. Thus, the top 20,000 Alexa sites have similar redirection characteristic as a random selection of 20,000 Alexa sites.

A negligible fraction of sites use HTTP header redirection; the malware and phishing sites are the most prominent source of such redirects, accounting for approximately 0.2%. The primary differentiator with respect to redirection are those sites employing meta refresh and JavaScript

Table 4.3: Average Redirection Chain Lengths

Host Population	VP1	VP2	VP3	VP14
Alexa Top 20K	1.45	1.44	1.44	1.44
Alexa Random 20K	1.33	1.33	1.33	1.32
Malware / Phishing	1.29	1.30	1.31	1.30
Spam	1.16	1.16	1.15	1.14

redirection. Meta refresh is a tag which is part of the HTML web page content rather than the HTTP protocol, while JavaScript is a scripting language supported by many browsers. The malware and phishing sites are more than twice as likely to use meta refresh and JavaScript as compared to legitimate Alexa sites.

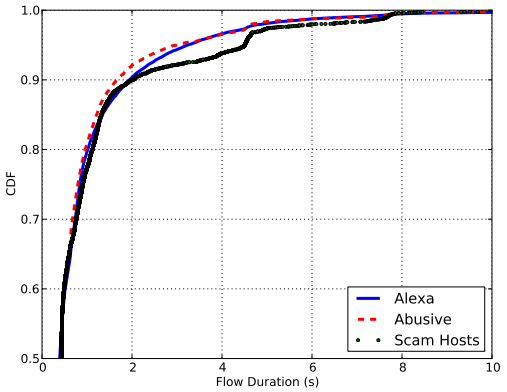
In addition, we looked at the average redirection chain length of each host population by vantage point. What we found there was not a large difference between each of the populations. Also, we found that legitimate URLs actually were redirected on average more often than abusive ones. Due to the small difference between average chain lengths, we believe the discriminatory power of chain lengths as a feature in our classification adds no value. Table 4.3 shows the average redirection per host population.

4.1.3 Netflow is too coarse-grained

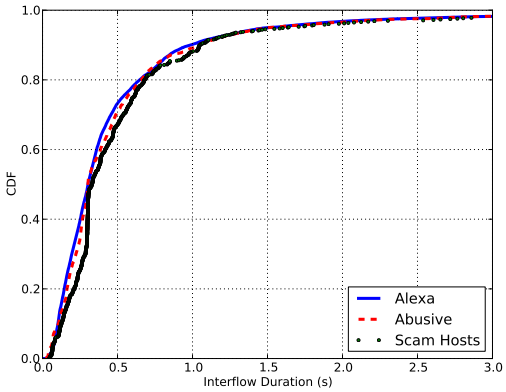
The three primary features in RBSeeker [5] and other redirection detection schemes are: i) flow duration; ii) inter-flow duration; and iii) flow size. The intuition is that redirection creates small and fast flows (e.g., a small redirection page or status code in the HTTP header) and are closely spaced in time (as the redirection is automated, rather than human driven).

These three features are readily available from Netflow records and RBseeker uses them in addition to other features, e.g., the DNS, to detect redirecting bots. To better understand the efficacy of Netflow features in this role, we examine the distribution of these features among abusive and legitimate sites based on our collected data.

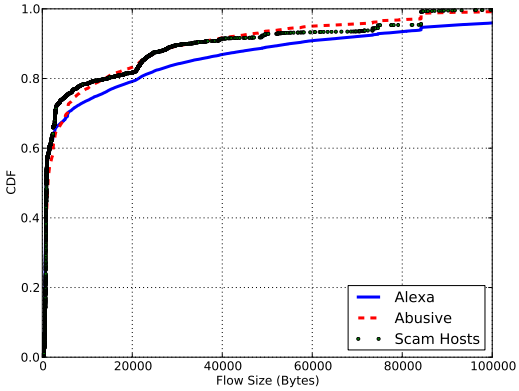
From the packet captures resulting from following URLs and fetching content from the sources detailed in Table 3.3, we generate Netflow records. We then examine the time duration and total bytes of the first flow for any given URL fetched. When a site has one or more redirections, we measure the inter-flow duration, or time between successive flows corresponding to an origin URL.



(a) Duration



(b) Interflow Duration



(c) Bytes

Figure 4.4: Distribution of features available from Netflow evident in our datasets. Flow duration, interflow duration, and flow size are largely similar, suggesting that Netflow features are too coarse-grained and not sufficiently discriminative.

Figure 4.4 depicts the cumulative fraction of each of these Netflow features as observed in our three datasets: the Alexa listed hosts, external abusive hosts, and our honeypot-gathered scam hosts. We see the total time duration (4.4(a)) and interflow durations (4.4(b)) are largely indistinguishable between legitimate and abusive sites. While the total number of bytes of Alexa listed hosts is generally larger and has a longer tail (4.4(c)), the results are again quite similar.

Redirection is only one potential means used to hide and protect abusive infrastructure a second is proxies. In addition to the relative lack of differentiation afforded by redirection, such techniques cannot detect proxies.

Thus, our fundamental contention is fine-grained transport traffic data offers specific benefits over the current practice of using Netflow records to detect abusive behavior, without compromising privacy or processing performance. We validate this assertion next.

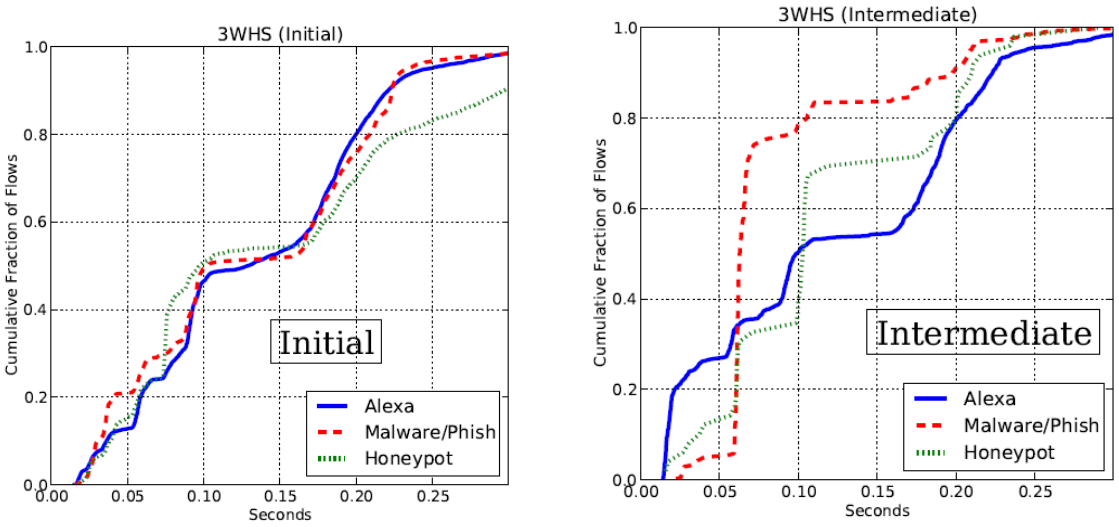
4.2 Discriminative Features

Next step we performed basic data exploration to understand which features are likely to be discriminative, as well as to substantiate or disprove our intuitions regarding abusive host traffic. To understand the utility of each different traffic feature in relation to the point along any redirection chain, we therefore divide our redirection chain analysis into: "initial," "intermediate," and "terminal," states corresponding to the first URL, the set of any intermediate URLs due to redirection, and the final landing page.

Timing, including initial RTT as inferred from the TCP three way handshake, as well as the RTT variance, inter-arrival time variance (jitter), and maximum idle time over the sequence of packets is one powerful aspect of our approach not available in coarse-grained Netflow. We first examine the distribution of initial RTTs among the different website populations in Figure 4.5.

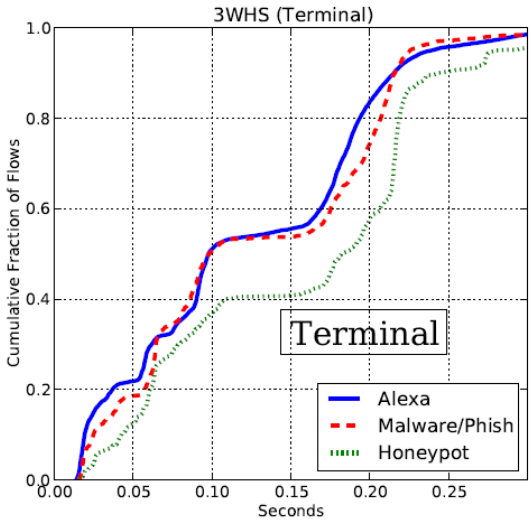
Somewhat surprisingly, we see little initial RTT difference between legitimate and abusive websites. To better understand this initial RTT phenomenon relative to our United States vantage points, we perform a basic continent geolocation of hosts in each population using MaxMind [35]. Table 4.4 shows two interesting facts: first, continental host distribution varies little among abusive and legitimate sites, suggesting location or RTT-based techniques cannot suffice to identify all abusive hosts. Second, North America dominates across all populations, with between 50-52% of the hosts. We chose to geolocate these websites because of how location, specifically the distance between a webserver and a requesting host, plays a role in a three-way handshake (3WHS) relative to our vantage points. 3WHS total time is one of our classification features we wanted to see if this feature's discriminatory power changes based on location of the abusive infrastructure.

In Figure 4.5 we look at the cumulative distribution of flows by their initial RTT of the 3WHS. We subdivided the distributions between initial, intermediate, and terminal. Initial refers to only the first URL in a redirection chain and is the one mined in the host collection process. If there is only one URL in the whole redirection chain then it would fall into this category. Intermediate refers to all of the flows between the initial and the final URL in the redirection chain and terminal is the final URL (not including the initial).



(a) Initial

(b) Intermediate



(c) Terminal

Figure 4.5: Distribution of initial RTT from TCP three way handshake among different website populations.

In Figure 4.5(a) we see little to no difference in the cumulative distribution between abusive and non-abusive flows. So at first glance we see no benefit from using initial RTT of the 3WHS for our host population in the determination of abusiveness. But if we analyze the whole redirection chain, not just the initial URL, we see large differences between the populations during the intermediate flows of the redirection as seen in Figure 4.5(b). This variation is most likely caused

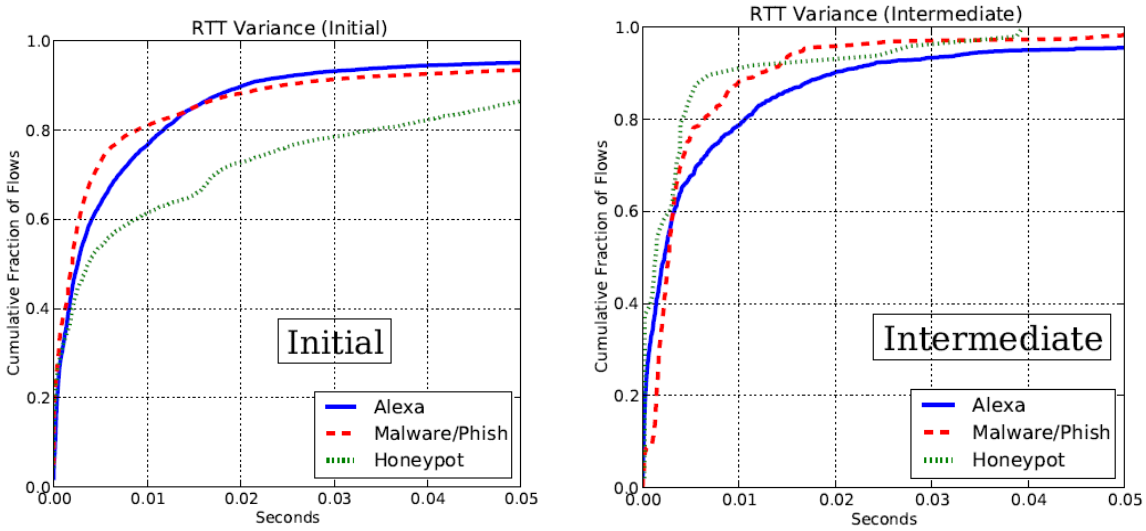
Table 4.4: Continental distribution of host populations

	Alexa		Abusive	
	Top	**Random**	**Honeypot**	**External**
NA	50.9	50.6	51.0	51.6
EU	29.5	34.2	35.4	31.1
AS	17.0	12.5	12.1	7.3
SA	1.6	1.6	0.3	7.3

by different hosts which supports different functionality. In the terminal plot (Figure 4.5(c)), we don't see as much variability between Malware/Phishing and Alexa but we do see a large difference between the Honeypot (spam) and the other populations. The differences found in the intermediate and terminal phases of the redirection chains allows the three-way handshake feature to be useful for classification.

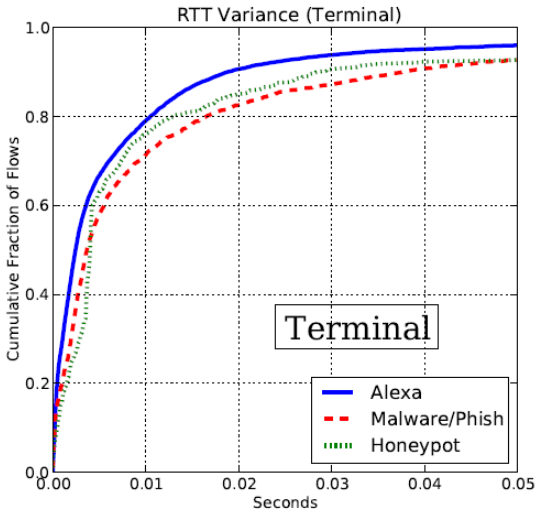
We next turn to the variance of RTT samples, as inferred by our technique. Figure 4.6 shows RTT variance is a strong indicator for abusive hosts of scam infrastructure as discovered by our honeypot. Like the plots in the three-way handshake, we sub-divided the plots for RTT variance into initial, intermediate, and terminal. While the difference and variability between the plots is not as significant is size as seen in the three-way handshake plots we still find discriminatory power in leveraging the RTT variance feature.

Looking at the initial plot in Figure 4.6(a) we see the Alexa and Malware/Phishing overlap consistently through their respective plots. But the Honeypot population shows a more distributive RTT variance. So in this instance the discriminatory power for the initial RTT variance is between the Honeypot and the rest of the population. In the intermediate phase, as shown in Figure 4.6(b), the Malware/Phishing and Honeypot populations seem more consistent with each other. While the Alexa population changes little from the initial plot. This is probably due to consistency in the resources available to non-abusive hosts. The interesting point of this plot is that almost 90% of the malicious flows have variance less than or equal to .01 seconds while the Alexa flows show only 80% at the same variance. Are initial assumption that abusive hosts use under-provisioned resources as compared to their counterparts seems to be contradicted here. This could be due to rate limiting or load balancing. We leave this investigation for future work. But none-the-less we can leverage this difference between the flows in our classification methodology. Finally in Figure 4.6(c) we see little variability in the three plots until we look at the final 40% of flows depicted. At this point the three plots start separating to their peak distribution at 80%. Again, this difference shows us there exists discriminatory value for this



(a) Initial

(b) Intermediate

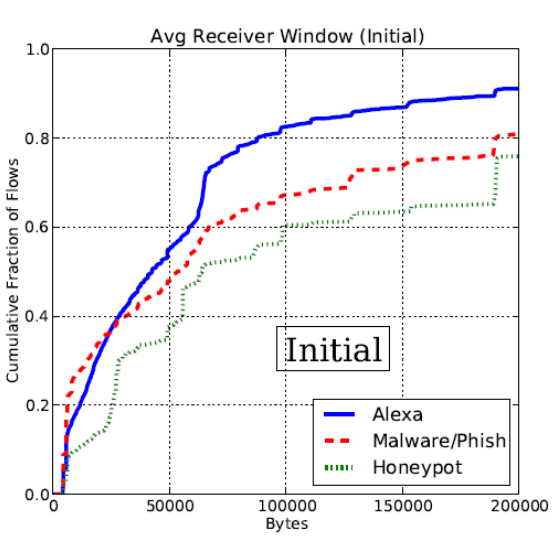


(c) Terminal

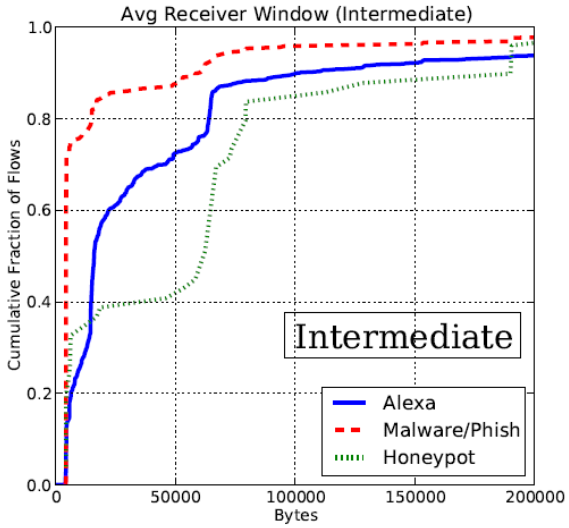
Figure 4.6: Distribution of RTT variance among different website populations.

portion of the redirection chain. At each plot we find evidence of how RTT variance adds value as a feature to our methodology.

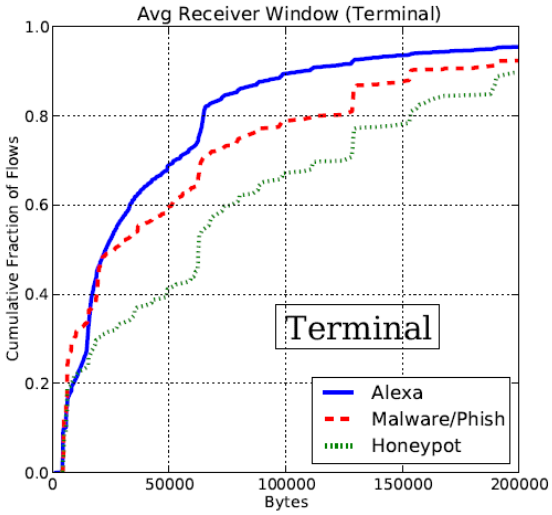
As a final example, we discover the receiver window is a strong differentiator of abusive versus legitimate hosts. Recall the TCP receiver window provides flow-control (which is different than congestion-control): a host which is unable to read from its local socket buffer quickly enough can slow the remote source to prevent buffer overrun.



(a) Initial



(b) Intermediate



(c) Terminal

Figure 4.7: Distribution of RTT variance among different website populations.

We collect the initial receiver window value, its average, maximum, and the number of times it goes to zero. All values are scaled appropriately given TCP window-scale options. Figure 4.7 shows the distribution of average receiver window sizes among the different populations for different stages of the redirection chains.

Perhaps counter-intuitively, the receiver window sizes are smaller for the legitimate Alexa web

sites. After a detailed exploration, we believe this phenomenon reveals it is due mainly in part to the advanced receiver window auto-tuning features present in modern server operating systems. Because legitimate servers are often quite busy, and are sending much more traffic than they are receiving, the operating system elects to allocate minimal buffer space to the socket to conserve resources. Further, the values which regulate receiver buffer sizes are a function of available network subsystem memory. Specifically, Linux exposes these parameters via tunable sysctls: `tcp_moderate_rcvbuf` (since Linux 2.4.17 and 2.6.7) and `tcp_rmem` (since Linux 2.4) [36]. Other operating systems perform similar optimizations.

Thus, the receiver window is a powerful method by which to identify busy, well-provisioned servers. The other receiver window features, including minimum and maximum also serve to identify properties of the end-system and the load it is under. For example, old operating systems – common among compromised residential hosts – do not perform any window scaling, and therefore are immediately identifiable by this feature alone.

4.3 Classification Performance Results

Given the encouraging results from the aforementioned discriminative features, we run two binary classification (Naïve Bayes and C4.5 Decision Trees), supervised learning algorithms across different combinations of legitimate and abusive hosts in our population (Table 3.3), for traffic as observed at our four vantage points (Table 3.1).

In assessing classification performance, all of the metrics in Section 3.5 are needed. For instance, accuracy is misleading if the underlying class prior is heavily skewed, for example if there are many more legitimate than abusive hosts being classified. Precision therefore measures, for hosts predicted as abusive, the fraction which are truly abusive, as determined by the population from which they originated. Recall measures the influence of misclassified hosts, therefore is a metric of the classifier's ability to detect abusive hosts. Specificity, or true negative rate, determines how well the classifier is differentiating between false positives and true negatives.

Our testing reveals the decision tree learner performs far better than the Naïve Bayes classification algorithm. Table 4.5 and Table 4.6 shows our performance results for different combinations of legitimate and abusive sites for vantage point four (VP4). All of the vantage points have similar results per classification algorithm. All results can be found in Appendix A.

We focus on the decision tree results. Our first observation is that transport features can identify

Table 4.5: Transport traffic classification performance - C4.5 Decision Tree

	Acc	**Pre**	**Rec**	**FPR**
Alexa vs. External	0.87	0.77	0.68	0.09
Alexa vs. Honeypot	0.94	0.84	0.81	0.03
Alexa Rnd vs. External	0.80	0.73	0.62	0.11
Alexa Rnd vs. Honeypot	0.94	0.85	0.80	0.03

Table 4.6: Transport traffic classification performance - Näive Bayes

	Acc	**Pre**	**Rec**	**FPR**
Alexa vs. External	0.74	0.57	0.70	0.24
Alexa vs. Honeypot	0.82	0.50	0.66	0.15
Alexa Rnd vs. External	0.73	0.57	0.65	0.23
Alexa Rnd vs. Honeypot	0.80	0.49	0.66	0.16

abusive hosts gathered from the honeypot better than the external malware and phishing sites. Whereas we achieve a 94% accuracy classifying abusive traffic from the honeypot sites, we only get 87% accuracy.

Looking at the false positive rates (FPR), we see that while the Honeypot population show a 3% rate, the external sites show 9% and 11% respectively. These rates are not insignificant especially when dealing with high volumes of traffic from large network connections like gigabit links. We leave the problem to decreasing the FPR to future work, but believe that approaches where we could merge our methodology to other existing approaches, like the ones discussed in Chapter 2, would produce a decreased FPR less than an either approach by itself.

Second, the Alexa top 20,000 represents well-provisioned hosts and sites supporting large volumes of traffic and commercial customers. As such, compared to a random sampling of 20,000 Alexa hosts, the top 20,000 provide a better basis for classifying legitimate hosts. As you approach the middle-tier to lower-tier of the Alexa Top 1 Million sites you start viewing sites not only limited in resources but also located in countries where the internet infrastructure may be out-dated and under-provisioned. For example looking at the top two sites as reported from the analytics collected from the last 3 months ending in Aug 2012, show `google.com` and `facebook.com` each represent 5.4% and 6.3% of *estimated percentage of global pageviews*. While the 500th position site is 0.00255% [26]. This tells us that visitors to sites in the bottom tiers of the Alexa listed hosts most likely have very little traffic. These sites are likely not creating a revenue stream and as such probably not associated with expensive high-end infrastructure.

Table 4.7: Transport traffic F-scores (harmonic mean of precision and recall) using decision trees across vantage points and combinations of legitimate and abusive sites

	VP1	**VP2**	**VP3**	**VP4**
Alexa vs. External	0.72	0.71	0.73	0.72
Alexa vs. Honeypot	0.83	0.81	0.82	0.83
Alexa Rnd vs. External	0.65	0.67	0.67	0.67
Alexa Rnd vs. Honeypot	0.81	0.81	0.81	0.83

Table 4.8: Transport traffic F-scores (harmonic mean of precision and recall) using Näive Bayes across vantage points and combinations of legitimate and abusive sites

	VP1	**VP2**	**VP3**	**VP4**
Alexa vs. External	0.64	0.63	0.63	0.63
Alexa vs. Honeypot	0.57	0.57	0.58	0.56
Alexa Rnd vs. External	0.63	0.61	0.61	0.61
Alexa Rnd vs. Honeypot	0.56	0.56	0.54	0.55

Next, we examine the classification F-scores for each of our four vantage points. Table 4.7 and Table 4.8 shows the F-scores for both decision trees and Näive Bayes classification algorithms. Again we see that the decision tree performs better, and we will refer to those results when discussing F-scores.

Because there is a natural tension between achieving high precision and high recall, the commonly used F-score metric takes the harmonic mean of precision and recall. Table 4.7 shows the F-scores are consistently better for the honeypot sites, across all vantage points. Further, performance for all datasets remains consistent across vantage points, implying the technique works well for a variety of different connection types and physical locations.

The reasons why the classifier performs better on the Honeypot sites than External ones should be investigated more in future work. The difference could be due to the type of compromised systems that exploit using spam versus other malicious behavior.

Thus, against traffic from the abusive hosts supporting websites as discovered in our honeypot, we reliably achieve greater than 90% accuracy with less than a 4% false positive rate. These results suggest transport traffic analysis can greatly aid existing schemes by providing a method which is largely orthogonal, hence boosting performance. Further, as previously discussed, transport traffic features are difficult for adversaries to subvert. We leave analysis of the combination of transport classification and existing schemes to future work.

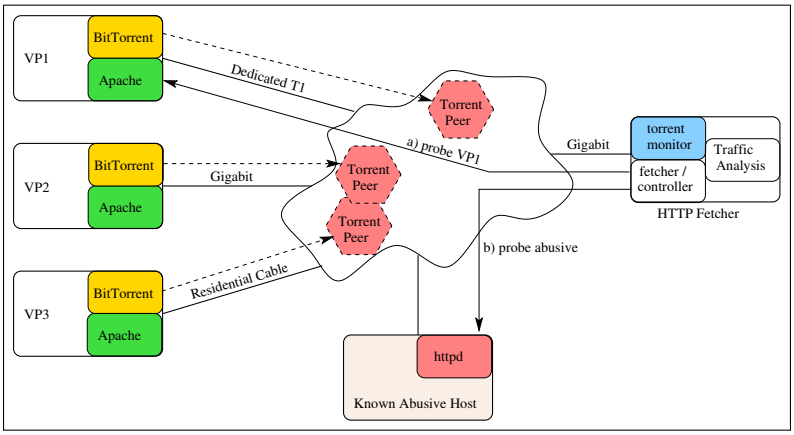


Figure 4.8: Test and Setup

4.4 Congestion Sensitivity

A natural concern with our approach is the traffic features we extract are often indicative of congestion, suggesting legitimate hosts experiencing congestion might be misclassified as abusive. To better understand the sensitivity of our technique, we analyze the traffic of legitimate hosts which are congested with controlled, real-world BitTorrent traffic. BitTorrent is attractive as it allows continuous testing to explore time-of-day effects, is bursty, and exchanges traffic with many peers simultaneously.

In simple terms, we setup three peer-to-peer servers hosting large gigabyte sized files. Then we served these files, mostly linux distributions, to the general public for download. Our goal was to congest the network uplink for each of these peer-to-peer servers. One of the advantages of using BitTorrent is it's ability to rate limit the the files being served to the peer-to-peer network. While congesting the uplinks, we simultaneously fetched both abusive and non-abusive URLs from the same infrastructure hosting the peer-to-peer servers. During the fetching process we captured the transport traffic streams. Next we can analyze the data and see how much congestion can disrupt our classification.

Specifically, we set up the experiment detailed in Figure 4.8. Our three vantage points are each configured with an instance of the Apache HTTP daemon and a BitTorrent client. Note, in contrast to our previous use of vantage points (VP1-3) to collect traffic, we now use these same hosts as candidate "legitimate" web hosts.

The BitTorrent client seeds many public, legal torrents. Recall each of the vantage points reside on networks with varying access connectivity, so we may test different real-world congestion scenarios.

A centralized controller performs two functions. First, it continuously monitors the aggregate upload rate of each of the vantage points, as this rate varies depending upon Torrent load. Second, it fetches HTTP content from both the vantage point servers as well as a list of known abusive hosts. We analyze the traffic features of the two classes: legitimate but congested flows, versus abusive flows.

Our classifier was trained just as in the main experiment. We took the flows from each of the three peer-to-peer vantage points and classified them as either abusive or not-abusive. Taking the same vector of features on a per-flow basis we trained the classifier. We also attempted the Support Vector Machine [13] model learning algorithm to see if we would get better results for our classifier, but the results show that the best algorithm was the C4.5 decision tree. Then we used the same 10-fold cross-validation test to determine accuracy and other prediction metrics (FPR, TN, TP, FN, FP).

Figure 4.9 depicts the transport traffic analysis performance, measured by F-score, as a function of background congestion and link connectivity. Note the physical link characteristics of VP1 and VP3, a dedicated T1 and residential cable respectively, place an upper-bound on our background Torrent upload rate. We were unable to saturate VP2, which has a Gigabit link.

Our classification performance remains high in the face of background traffic. In addition performance appears to be insensitive to the intensity of background traffic. Across all background traffic levels, we continue to see receiver window features as most discriminatory, as we discussed in Section 4.2 receiver window features are insensitive to background cross-traffic. However, we find other discriminative features involved in the classification change as a function of the background traffic – emphasizing the power of the statistical learner: as one feature yields less power, others provide more information. For instance, as background traffic increases, the maximum idle time, jitter, and RTT variance become more important in the classification decision. As an example, we show in Figure 4.9 the features that provided the highest discriminatory power at particular stage of the congestion experiment for VP3. Our statistical learner uses the Relief Algorithm [37] when determining the weight of each feature during classification. The data for the other vantage points are found in Appendix B.

Table 4.9: VP3 - Top 5 Features Used for Classification at Specified Upload Rates

	100KB/s	**250KB/s**	**500KB/s**
#1	Min Receiver Window	Min Receiver Window	Min Receiver Window
#2	3WHS	Total Duration	Total IDLE Time
#3	# SRC PKTS	# SRC PKTS	Total Duration
#4	Total Duration	Total IDLE Time	Jitter
#5	# DST PKTS	3WHS	# SRC PKTS

Finally, we examine the effect of the fetched object size and time-of-day. While we might expect worse classification performance when larger objects are fetched in the presence of background traffic, as doing so might exacerbate the congestion, we find the classification performance does not depend on the object size. Similarly, overall network traffic is well-known to vary as a function of time-of-day [38]. Again, we see the set of discriminative features change as a function of time-of-day. For example, the total number of retransmissions is much more powerful during the evening, off-peak hours as compared to peak hours. However, we do not observe the classification performance varying as a function of the hour of the day the flows are collected, as shown in Figure 4.10.

4.4.1 Limitations

Even though we were able to show that background congestion can be overcome there are still limitations to be explored. First of all we looked only at host populations from the spam Honeypot and only fetched them from a controlled environment. For the experiment to be more robust the need to incorporate a more wide range of known abusive URLs exists. In addition, BitTorrent, while useful in congesting a machine's local uplink is probably not fully representative of most congestion. Adding combinations of other types of network traffic along with peer-to-peer traffic would add more of a real-world effect to the experiment. Having a mixture of large communication streams from peer-to-peer sessions along with smaller sessions like HTTP requests would inject more burstiness and randomness. Finally, adding more congested serving nodes beyond three would also allow for more robust data collection.

4.5 Overhead

Finally, we examine the performance impact of our transport traffic collection agent, as compared to industry-standard Netflow. That is does our collection agent inject an unreasonable amount of overhead in processing time or resource usage as compared to Netflow? The performance impact is the delta between how long it takes our agent to process the same information

as compared to Netflow. To mimic the overhead of Netflow, we employ two open-source tools: `flowtools` as a Netflow collection agent, and `softflowd` [39] to read a pcap capture file, create flows, and export them to the flowtool collector [40]. We use the same real-world pcap capture file for testing both Netflow and our collection agent. The file contains approximately 2M packets, gathered over 10 hours, with an average flow of 55 bytes taking 3.3 seconds. For each tool, we use the UNIX `time` utility to measure total time elapsed and estimate computational overhead.

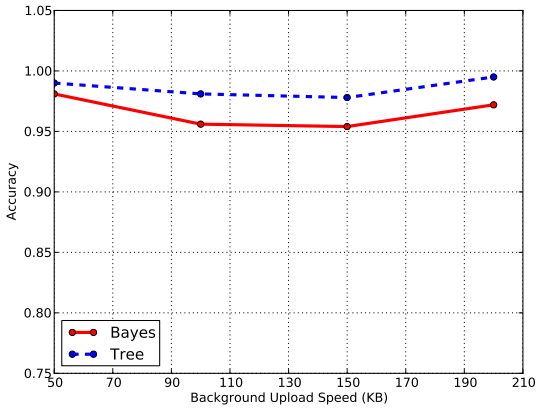
Our transport traffic analysis capture requires 1.83s, while `softflowd` takes 1.13s. The `flowtools` capture program uses negligible resources, taking only 0.06s of system time. Thus, we approximate total time for Netflow as the sum: 1.19s. Extrapolating these numbers, our un-optimized collection agent can process more than 1M packets per second. With further optimization, and dedicated hardware support, we believe much higher packet rates are possible.

Thus, although per-packet captures are considered infeasible due to the large storage overhead, the extra computational cost of maintaining the additional fine-grained features we leverage in this work is minimal.

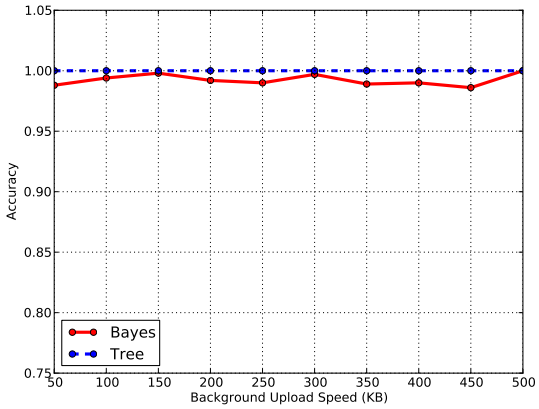
Our tests were done in software only. The next step would be to incorporate our collection agent in hardware like a router. We leave this for future work, but believe that our approach would work in hardware with minimal added overhead.

4.6 Conclusion

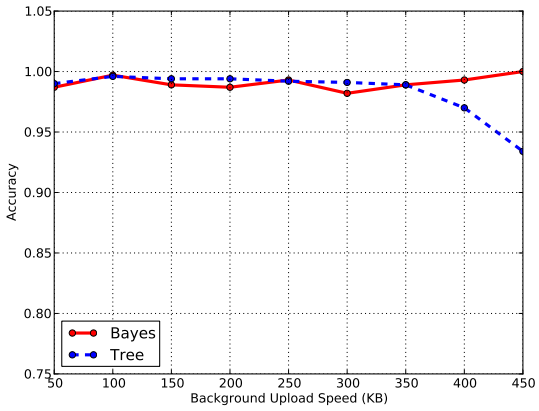
In this chapter we discussed the results of our approach to the detection of abusive infrastructure. Our results show we were able to successfully classify malware/phishing flows with an accuracy rate as high as 87% and classify spam flows with an accuracy rate as high as 94%. We have shown three examples (RTT Variance, 3WHS timing, and receiver window size) of how are feature selection can provide discriminatory power. In addition, we compared and contrasted our collection agent to Netflow and discussed why a finer-grained collection method is helpful in determining abusiveness. Finally, we showed results of our congestion experiments and overhead comparisons. A full break down of classification analysis results can be found in Appendix A. Results for the congestion experiment can be found in Appendix B.



(a) VP1 (T1)



(b) VP2 (Gigabit)



(c) VP3 (Cable)

Figure 4.9: Transport traffic analysis performance as a function of background congestion and link connectivity.

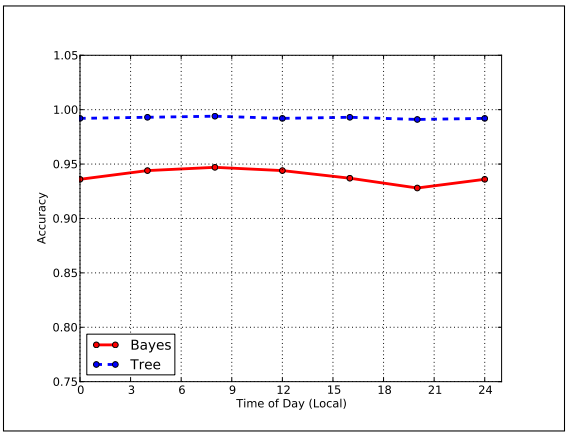


Figure 4.10: Transport traffic analysis performance as a function of time-of-day

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 5:
Conclusion and Future Work

5.1 Conclusion

In this thesis we hypothesized it is possible to characterize abusive infrastructure given transport features gathered on a per-flow basis from TCP headers and packet timing. To investigate this hypothesis, we explored a passive traffic analysis approach for detecting abusive infrastructure relying on content inspection, sender reputation, or command and control signatures. We present a novel solution based on fine-grained transport traffic features.

Our results demonstrate that per-packet TCP header and timing information within a network flow, in conjunction with supervised learning techniques, can discriminate known abusive traffic from known good traffic, and hence suggests at variety of means for detecting abusive infrastructure. Further, our IP address and content agnostic approach can be more widely deployed than previous techniques, for instance in networks or countries with strict privacy constraints.

The full list of features are provided in Section 3.4. Our results show that to achieve an accuracy rate as high as 94% for URLs collected by our honeypot and 87% for malware/phishing URLs collected from two well-known blocklists. Future work should include incorporating more data from a wider range of sources including live traffic.

In addition, we looked at redirection as a possible indicator for abusive behavior e.g., number of redirections per requested URL. Somewhat surprisingly, we found that per-device redirection and general redirection is common for legitimate websites. Looking at a per-device perspective we found approximately 20% of Alexa listed sites employ at least one user-agent specific redirection. We also surprisingly found that legitimate websites redirected approximately 12% more than the spam, malware, or phishing sites. What we found was that the variability in the average redirection chain size was too small to use as a feature. The average legitimate chain length was 1.45 while spam and malware/phishing sites were of average direction length 1.29 and 1.16 respectively. These experiments are detailed in Section 4.1.1.

An initial concern we had with our approach is that the features we extract are often indicative of congestion. To explore the concern that legitimate hosts experiencing congestion may be classified as abusive we conducted an experiment to analyze congested traffic. Using real-

world BitTorrent traffic to congest the network uplink of three webservers under our control, we conducted fetches of URL content from legitimate and abusive sites. Collecting our features and training our supervised learning classifier as we did for our main experiment, we still averaged greater than 90% accuracy differentiating abusive from good sites.

Finally, we examined the overhead of our feature collection agent compared to the features available from the widely-deployed and standardized Netflow. Speed and resource usage is a key component when running any system on a real-time network. To see how our system performed, we compared it to `flowtools` [40], a Netflow collection agent, and `softflowd` [39], a program that exports flows to the collection agent. We found in our experiment on 2M packets that our system took 1.83 seconds as compared to 1.19 seconds for the Netflow system to extract flow features.

5.2 Potential Mitigation Strategies for the Adversary

The transport traffic features we explore are low-level: some are inherent to the environment, such as cross-traffic, packet reordering, and timing, while others are specific to the end-system, including receiver window behavior, and congestion. While not impossible for adversaries to modify their flow behavior in a manner to evade detection, our technique presents several impediments to doing so.

First, an abusive host might attempt to measure its available uplink bandwidth in order to rate limit its traffic. However, the dynamic properties of the network, and high-levels of statistical multiplexing on commodity connections, imply such efforts will likely be unreliable.

Second, other aspects, such as receiver window behavior require operating system and kernel modifications and as such are hard to effect. If an adversary does control the host they maybe able to set some of the receiver window settings, but only up to the point the operating system supports. In particular, closed-source systems provide few methods to modify the core TCP/IP stack. If the operating system does optimize receiver windows as a function of available memory, then there is little an owner can do to get around the behavior itself, thereby making the feature valuable for our discrimination.

Third, "low-and-slow" type of stealth strategies [5] may be effective in evading our transport traffic technique, but impart a significant abusive infrastructure cost as more systems are needed to achieve equivalent aggregate rates.

The bottom line is that for spam propagation, phishing attacks, and malware distribution to be successful, bandwidth, speed, and host resources are needed. To mask their traffic successfully, malicious hosts must mimic average network traffic. By doing so, though, they significantly decrease their ability to economically launch successful campaigns.

5.3 Future Work

The following section outlines future work crucial to abusive infrastructure detection as it relates to the research performed in this thesis.

Bot Detection

A natural question is if our proposed method can detect not just abusive infrastructure, but bots and botnets specifically. Across our malicious host populations, we perform reverse DNS (PTR record) lookups. For each, we use a set of heuristics to determine whether the DNS name corresponds to a residential connection or not. We use an existing tool containing a list of keyword heuristics, e.g., "cable", "dsl".

Surprisingly, only 2% of the unique IPs are residential based on the DNS name heuristics. Approximately 37% have no DNS PTR record at all. The remaining 61% are non-residential. We then use our classifiers to attempt to differentiate between residential and non-residential abusive hosts. While we achieve very high accuracy, approximately 97% using decision trees and Naïve Bayes, this level of accuracy is due only to the large class imbalance. The F-score is poor, at only .09. We leave the investigation of finding bots among the abusive hosts for future work.

Support Vector Machine

We tested both Naïve Bayes and C4.5 decision tree learning algorithms when creating our classifier. We found that the decision tree provided the highest level of accuracy with a lower false positive rate than Naïve Bayes. We propose that future work using our classification model incorporate other learning algorithms, specifically including Support Vector Machines (SVM) [13]. Instead of using strictly linear classification, SVM can classify two-class samples by constructing hyperplanes in a multidimensional space. These spaces separate binary classes. By using other algorithms, and trying out other variables (e.g., different kernel functions for SVM) via grid search we may be able to decrease the false positive rate and increase accuracy even further.

Resource Contention

While this paper does not focus on network congestion or other forms of resource contention, we recognize that host processing load may have a significant impact on several transport traffic characteristics such as the initial RTT and RTT variance. Intuitively, most Alexa web sites should experience higher levels of server load than malicious ones because of their much larger customer bases. This may explain why the abusive hosts exhibit better initial RTTs at the beginning and middle stages of the redirection chain, as illustrated in Figure 4.5. Further experiments are needed to better understand this phenomenon.

Real-World Congestion

To increase the scope of our experimentation we would also like to include real background traffic such as from our own campus being an immediate initial target. Using traffic generated by real users would inject actual internet user clutter into the data captures. We will be able to investigate to what degree our classification approach is influenced by natural network behavior.

Blocklist Comparison

To see if our approach can find abusive hosts that have not been collected by major blocklists such as those used in this thesis (`phishtank` [25] or `malwaredomains` [23]). Finding such hosts, that can be verified as abusive, will show that our approach provides a natural compliment to techniques used by major blocklisting enterprises. Testing both techniques in conjunction with each other may result in a more robust system then either of the techniques individually.

Decrease False Positive Rates

While our results are quite encouraging, we would also like to achieve a lower false positive rate. Currently we have a false positive rate at 3% against the Honeypot and 9%–11% for the malware/phishing data set. As fraction this is a low false-positive rate, but can unfortunately result in a large absolute number of false-positives.

A way to achieve a lower false-positive rate would be to incorporate a robust whitelist of legitimate domains. Currently we use a whitelist of approximately 50 domains to compare URLs returned from the Honeypot and redirection chains. Our approach labels all flows coming from the Honeypot as abusive, and if the first URL in a redirection chain is labeled abusive then all of the flows in that chain are labeled abusive, if not in the whitelist.

A larger list should increase classifier accuracy.

Combining Methods

Additionally, we plan to study the power of combining methods, for instance using transport features as part of a multi-stage hypothesis test. We note many of the flows we examine are short-lived, consisting of few packets and few total RTTs. We wish to explore strategic use of the TCP maximum segment size variable, along with other low-level TCP mechanisms, to gather more packets across longer time scales in order to improve our classification performance. We also hope to apply the transport traffic analysis method to detecting automated attacks against legitimate web sites, including CAPTCHA solvers, vulnerability scans, etc.

Finally, we wish to deploy our technique in a real world environment. The results of such live testing will hence further validate and refine our methodology for prediction and calculate actual resource overhead, and permit comparison against other approaches.

5.4 Summary

This thesis shows that it is indeed possible to characterize abusive infrastructure by mining features of the packet transport header and packet timings within a flow in conjunction with supervised learning. The summary of our experimental results are listed below:

- Looking at redirection types we saw legitimate sites redirected using 3xx status codes 87% of the time as compared to 73% from malware and 88% from the honeypot listed sites.

- We conducted an overhead experiment to see how our collection agent compared to Netflow overhead. Using 2M packets as our dataset our agent took 1.83 seconds and the Netflow agent took 1.19 seconds.

- From our dataset we found that continental host distribution varies little among abusive and legitimate sites. Also, North America dominates across all of our population sets with 50–52% of the hosts.

- Average redirection chain lengths for both legitimate and abusive sites were fairly close. The Alexa listed hosts averaged 1.30, malware/phishing sites averaged 1.29, and spam averaged 1.16.

- We found that per-device redirection is common. An investigation of the top 20,000 Alexa listed sites show that at least 20% employ at least one type of user-agent redirection. In addition, 40% of the sites return different content depending on advertised browser.

- We found that general redirection is common. Using only one advertised browser type, we found that legitimate sites (from Alexa) redirected an average of 30% and abusive sites an average of 18%.

- Using the features from Netflow as described by RB-Seeker's [5] experiment (duration, interflow duration, and bytes) we found that using our datasets the distribution of legitimate and abusive flows were not sufficiently discriminative. This tells us that Netflow features are too coarse-grained to be used in our classification approach.

- Our results were similar across all of our vantage points. Our f-score calculations show that the variability was no larger than .02 across any vantage by host population.

- The experiment to see if our approach will classify a legitimate host as abusive, due to high levels of congestion, resulted in a classification performance accuracy greater than 90%.

- Our methodology detects 94% for URLs collected by our honeypot and 87% for malware/phishing URLs collected from two well-known blocklists. The honeypot results show a false positive rate of 3% and 9–11% for malware and phishing URLs. These results were found using the C4.5 decision tree classification algorithm.

While the long-term effect of our technique on the adversary remains to be seen, we anticipate our system having positive net benefit in the security arms race.

APPENDIX A:
Classification Analysis Results

Vantage Point #1: hermes.cmand.org

Table A.1: VP1 Alexa Vs Externals: Confusion Matrix

Method	TP	FP	FN	TN
Bayes	6186	4407	2531	14996
Decision Tree	6044	2032	2673	17371

Table A.2: VP1 Alexa Vs Externals: Accuracy Results

Method	Accuracy	Standard Error
Bayes	0.753	0.004
Decision Tree	0.833	0.002

Table A.3: VP1 Alexa Vs Externals: Statistics

Method	Sensitivity	Specificity	PPV	NPV
Bayes	0.710	0.773	0.584	0.856
Decision Tree	0.693	0.895	0.748	0.867

Table A.4: VP1 Alexa Vs Spam: Confusion Matrix

Method	TP	FP	FN	TN
Bayes	2869	2906	1438	16497
Decision Tree	3495	621	812	18782

Table A.5: VP1 Alexa Vs Spam: Accuracy Results

Method	Accuracy	Standard Error
Bayes	0.817	0.004
Decision Tree	0.940	0.001

Table A.6: VP1 Alexa Vs Spam: Statistics

Method	Sensitivity	Specificity	PPV	NPV
Bayes	0.666	0.850	0.497	0.920
Decision Tree	0.811	0.968	0.849	0.959

Table A.7: VP1 Random Alexa Vs External: Confusion Matrix

Method	TP	FP	FN	TN
Bayes	5910	4222	2807	14243
Decision Tree	5090	1765	3627	16700

Table A.8: VP1 Random Alexa Vs External: Accuracy Results

Method	Accuracy	Standard Error
Bayes	0.741	0.002
Decision Tree	0.802	0.002

Table A.9: VP1 Random Alexa Vs External: Statistics

Method	Sensitivity	Specificity	PPV	NPV
Bayes	0.678	0.771	0.583	0.835
Decision Tree	0.584	0.904	0.743	0.822

Table A.10: VP1 Random Alexa Vs Spam: Confusion Matrix

Method	TP	FP	FN	TN
Bayes	2820	3034	1487	15431
Decision Tree	3406	666	901	17799

Table A.11: VP1 Random Alexa Vs Spam: Accuracy Results

Method	Accuracy	Standard Error
Bayes	0.801	0.002
Decision Tree	0.931	0.002

Table A.12: VP1 Random Alexa Vs Spam: Statistics

Method	Sensitivity	Specificity	PPV	NPV
Bayes	0.655	0.836	0.482	0.912
Decision Tree	0.791	0.964	0.836	0.952

Vantage Point #2: ralph.cmand.org

Table A.13: VP2 Alexa Vs Externals: Confusion Matrix

Method	TP	FP	FN	TN
Bayes	6145	4685	2660	14826
Decision Tree	5971	1793	2834	17718

Table A.14: VP2 Alexa Vs Externals: Accuracy Results

Method	Accuracy	Standard Error
Bayes	0.741	0.003
Decision Tree	0.837	0.002

Table A.15: VP2 Alexa Vs Externals: Statistics

Method	Sensitivity	Specificity	PPV	NPV
Bayes	0.698	0.760	0.567	0.848
Decision Tree	0.678	0.908	0.769	0.862

Table A.16: VP2 Alexa Vs Spam: Confusion Matrix

Method	TP	FP	FN	TN
Bayes	2903	2886	1457	16625
Decision Tree	3533	665	827	18846

Table A.17: VP2 Alexa Vs Spam: Accuracy Results

Method	Accuracy	Standard Error
Bayes	0.818	0.004
Decision Tree	0.937	0.002

Table A.18: VP2 Alexa Vs Spam: Statistics

Method	Sensitivity	Specificity	PPV	NPV
Bayes	0.666	0.852	0.501	0.919
Decision Tree	0.810	0.966	0.842	0.958

Table A.19: VP2 Random Alexa Vs External: Confusion Matrix

Method	TP	FP	FN	TN
Bayes	5736	4295	3069	14209
Decision Tree	5446	2057	3359	16447

Table A.20: VP2 Random Alexa Vs External: Accuracy Results

Method	Accuracy	Standard Error
Bayes	0.730	0.002
Decision Tree	0.802	0.002

Table A.21: VP2 Random Alexa Vs External: Statistics

Method	Sensitivity	Specificity	PPV	NPV
Bayes	0.651	0.768	0.572	0.822
Decision Tree	0.619	0.889	0.726	0.830

Table A.22: VP2 Random Alexa Vs Spam: Confusion Matrix

Method	TP	FP	FN	TN
Bayes	2878	2998	1482	15506
Decision Tree	3499	619	861	17885

Table A.23: VP2 Random Alexa Vs Spam: Accuracy Results

Method	Accuracy	Standard Error
Bayes	0.804	0.002
Decision Tree	0.935	0.002

Table A.24: VP2 Random Alexa Vs Spam: Statistics

Method	Sensitivity	Specificity	PPV	NPV
Bayes	0.660	0.838	0.490	0.913
Decision Tree	0.803	0.967	0.850	0.954

Vantage Point #3: bob.cmand.org

Table A.25: VP3 Alexa Vs Externals: Confusion Matrix

Method	TP	FP	FN	TN
Bayes	6490	4817	2670	14589
Decision Tree	6431	2140	2729	17266

Table A.26: VP3 Alexa Vs Externals: Accuracy Results

Method	Accuracy	Standard Error
Bayes	0.738	0.001
Decision Tree	0.830	0.002

Table A.27: VP3 Alexa Vs Externals: Statistics

Method	Sensitivity	Specificity	PPV	NPV
Bayes	0.709	0.752	0.574	0.845
Decision Tree	0.702	0.890	0.750	0.864

Table A.28: VP3 Alexa Vs Spam: Confusion Matrix

Method	TP	FP	FN	TN
Bayes	3058	3066	1329	16340
Decision Tree	3504	633	883	18773

Table A.29: VP3 Alexa Vs Spam: Accuracy Results

Method	Accuracy	Standard Error
Bayes	0.815	0.002
Decision Tree	0.936	0.002

Table A.30: VP3 Alexa Vs Spam: Statistics

Method	Sensitivity	Specificity	PPV	NPV
Bayes	0.697	0.842	0.499	0.925
Decision Tree	0.799	0.967	0.847	0.955

Table A.31: VP3 Random Alexa Vs External: Confusion Matrix

Method	TP	FP	FN	TN
Bayes	5903	4447	3257	14232
Decision Tree	5583	1969	3577	16710

Table A.32: VP3 Random Alexa Vs External: Accuracy Results

Method	Accuracy	Standard Error
Bayes	0.723	0.002
Decision Tree	0.801	0.002

Table A.33: VP3 Random Alexa Vs External: Statistics

Method	Sensitivity	Specificity	PPV	NPV
Bayes	0.644	0.762	0.570	0.814
Decision Tree	0.609	0.895	0.739	0.824

Table A.34: VP3 Random Alexa Vs Spam: Confusion Matrix

Method	TP	FP	FN	TN
Bayes	2877	3386	1510	15293
Decision Tree	3426	670	961	18009

Table A.35: VP3 Random Alexa Vs Spam: Accuracy Results

Method	Accuracy	Standard Error
Bayes	0.788	0.003
Decision Tree	0.929	0.002

Table A.36: VP3 Random Alexa Vs Spam: Statistics

Method	Sensitivity	Specificity	PPV	NPV
Bayes	0.656	0.819	0.459	0.910
Decision Tree	0.781	0.964	0.836	0.949

Vantage Point #4: woland.cmand.org

Table A.37: VP4 Alexa Vs Externals: Confusion Matrix

Method	TP	FP	FN	TN
Bayes	6164	4715	2663	14757
Decision Tree	5929	1887	2898	17585

Table A.38: VP4 Alexa Vs Externals: Accuracy Results

Method	Accuracy	Standard Error
Bayes	0.739	0.003
Decision Tree	0.831	0.003

Table A.39: VP4 Alexa Vs Externals: Statistics

Method	Sensitivity	Specificity	PPV	NPV
Bayes	0.698	0.758	0.567	0.847
Decision Tree	0.672	0.903	0.759	0.859

Table A.40: VP4 Alexa Vs Spam: Confusion Matrix

Method	TP	FP	FN	TN
Bayes	2726	2821	1462	16651
Decision Tree	3274	676	914	18796

Table A.41: VP4 Alexa Vs Spam: Accuracy Results

Method	Accuracy	Standard Error
Bayes	0.819	0.003
Decision Tree	0.933	0.002

Table A.42: VP4 Alexa Vs Spam: Statistics

Method	Sensitivity	Specificity	PPV	NPV
Bayes	0.651	0.855	0.491	0.919
Decision Tree	0.782	0.965	0.829	0.954

Table A.43: VP4 Random Alexa Vs External: Confusion Matrix

Method	TP	FP	FN	TN
Bayes	5812	4367	3015	14222
Decision Tree	5414	1940	3413	16649

Table A.44: VP3 Random Alexa Vs External: Accuracy Results

Method	Accuracy	Standard Error
Bayes	0.731	0.002
Decision Tree	0.805	0.002

Table A.45: VP4 Random Alexa Vs External: Statistics

Method	Sensitivity	Specificity	PPV	NPV
Bayes	0.658	0.765	0.571	0.825
Decision Tree	0.613	0.836	0.736	0.830

Table A.46: VP4 Random Alexa Vs Spam: Confusion Matrix

Method	**TP**	**FP**	**FN**	**TN**
Bayes	2719	2979	1469	15610
Decision Tree	3269	669	919	17920

Table A.47: VP4 Random Alexa Vs Spam: Accuracy Results

Method	**Accuracy**	**Standard Error**
Bayes	0.805	0.002
Decision Tree	0.930	0.002

Table A.48: VP4 Random Alexa Vs Spam: Statistics

Method	**Sensitivity**	**Specificity**	**PPV**	**NPV**
Bayes	0.649	0.840	0.477	0.914
Decision Tree	0.781	0.964	0.830	0.951

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B:
Congestion Experiment Results

Table B.1: Results from Congestion Experiment

Method	**TP**	**FP**	**FN**	**TN**
Bayes	23664	306	3526	26884
Decision Tree	27066	95	124	27095
SVM	21868	0	5322	27190

Table B.2: Congestion Experiment: Accuracy Results

Method	**Accuracy**	**Standard Error**
Bayes	0.930	0.002
Decision Tree	0.996	0.000
SVM	0.902	0.001

Table B.3: VP1 - Top 5 Features Used for Classification at Specified Upload Rates

	50KB/s	**150KB/s**	**>200KB/s**
#1	Min Receiver Window	Min Receiver Window	Min Receiver Window
#2	3WHS	3WHS	# SRC PKTS
#3	# SRC PKTS	# SRC PKTS	3WHS
#4	# DST PKTS	# DST PKTS	# DST PKTS
#5	Total Duration	Total Duration	# DST Retransmits

Table B.4: VP2 - Top 5 Features Used for Classification at Specified Upload Rates

	100KB/s	**250KB/s**	**>500KB/s**
#1	Min Receiver Window	Min Receiver Window	Min Receiver Window
#2	3WHS	RTT	3WHS
#3	# SRC PKTS	Total IDLE Time	Total IDLE Time
#4	Total Duration	Total Duration	Total Duration
#5	# DST PKTS	3WHS	RTT

Table B.5: VP3 - Top 5 Features Used for Classification at Specified Upload Rates

	100KB/s	**250KB/s**	**>500KB/s**
#1	Min Receiver Window	Min Receiver Window	Min Receiver Window
#2	3WHS	Total Duration	Total IDLE Time
#3	# SRC PKTS	# SRC PKTS	Total Duration
#4	Total Duration	Total IDLE Time	Jitter
#5	# DST PKTS	3WHS	# SRC PKTS

REFERENCES

[1] K. Levchenko, A. Pitsillidis, N. Chachra, B. Enright, M. Félegyházi, C. Grier, T. Halvorson, C. Kanich, C. Kreibich, H. Liu, D. McCoy, N. Weaver, V. Paxson, G. M. Voelker, and S. Savage, "Click trajectories: End-to-end analysis of the spam value chain," in *Proceedings of the 2011 IEEE Symposium on Security and Privacy*, SP '11, (Washington, DC, USA), pp. 431–446, IEEE Computer Society, 2011.

[2] R. Beverly and K. Sollins, "Exploiting transport-level characteristics of spam," in *Proceedings of the Fifth Conference on Email and Anti-Spam (CEAS)*, Aug. 2008.

[3] G. Kakavelakis, R. Beverly, and J. Young, "Auto-learning of SMTP TCP Transport-Layer Features for Spam and Abusive Message Detection," in *Proceedings of the 25th USENIX Large Installation Systems Administration Conference (LISA)*, Dec. 2011.

[4] T. Ouyang, S. Ray, M. Rabinovich, and M. Allman, "Can network characteristics detect spam effectively in a stand-alone enterprise?," in *Proceedings of the 12th Conference on Passive and Active Network Measurement*, Mar. 2011.

[5] X. Hu, M. Knysz, and K. G. Shin, "Rb-seeker: Auto-detection of redirection botnets," in *Proceedings of 16th Annual Network and Distributed System Security Symposium*, 2009.

[6] B. Claise, "Cisco Systems NetFlow Services Export Version 9." RFC 3954 (Informational), Oct. 2004.

[7] J. Fontelles and H. Winkler, "Directive 2006/24/EC," *Official Journal of the European Union*, vol. L, no. 105, pp. 54–63, 2006.

[8] L. Feiler, "The legality of the data retention directive in light of the fundamental rights to privacy and data protection," *European Journal of Law and Technology*, vol. 1, no. 3, 2010.

[9] D. S. Anderson, C. Fleizach, S. Savage, and G. M. Voelker, "Spamscatter: characterizing internet scam hosting infrastructure," in *Proceedings of 16th USENIX Security Symposium*, (Berkeley, CA, USA), pp. 10:1–10:14, USENIX Association, 2007.

[10] G. Gu, J. Zhang, and W. Lee, "BotSniffer: Detecting botnet command and control channels in network traffic," in *Proceedings of the 15th Annual Network and Distributed System Security Symposium (NDSS'08)*, February 2008.

[11] Y. Xie, F. Yu, K. Achan, R. Panigrahy, G. Hulten, and I. Osipkov, "Spamming botnets: signatures and characteristics," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 171–182, 2008.

[12] P. Mockapetris, "Domain Names - Implementation and Specification." RFC 1035 (Standard), Nov. 1987.

[13] C. Cortes and V. N. Vapnik, "Support vector networks," vol. 20, pp. 1–25, 1995.

[14] Y. Zhao, Y. Xie, F. Yu, Q. Ke, Y. Yu, Y. Chen, and E. Gillum, "Botgraph: large scale spamming botnet detection," in *NSDI'09: Proceedings of the 6th USENIX symposium on Networked systems design and implementation*, pp. 321–334, 2009.

[15] G. Gu, R. Perdisci, J. Zhang, and W. Lee, "Botminer: Clustering analysis of network traffic for protocol- and structure-independent botnet detection," in *USENIX Security Symposium*, pp. 139–154, 2008.

[16] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *Proceeding os fhte 2010 IEEE Symposium on Security and Privacy*, (Washington, DC, USA), IEEE Computer Society, 2010.

[17] S. Russell and P. Norvig, "Artificial intelligence a modern approach," pp. 495–503, Pearson Education Inc, March 2010.

[18] E. Alpaydin, "Introduction to machine learning," pp. 17–38, 173–196, 331, MIT Press, 2004.

[19] I. Androutsopoulos, K. Koutsias, V. Chandrinos, G. Paliouras, and C. Spyropoulos, "An evaluation of naive bayesian anti-spam filtering," in *Proceeding of the Workshop of Machine Learning in the New Information Age*, pp. 9–17, 2000.

[20] E. Blanzieri and A. Bryl, "A survey of learning-based technique of email spam filtering," in *Artifical Intelligence Review*, vol. 29, pp. 63–92, March 1993.

[21] J. R. Quinlan, "C4.5: Programs for machine learning," Morgan Kaufmann Publishers, 1993.

[22] J. R. Quinlan, "Induction of decision trees," in *Machine Learning 1*, pp. 81–106, Kluwer Academic Publishers, 1986.

[23] "Malware Domain Black Hole DNS Sinkhole," 2012. `http://www.malwaredomains.com/`.

[24] "Malware domain list," 2012. `http://www.malwaredomainlist.com/`.

[25] "Phish tank," 2012. `http://www.phishtank.com/`.

[26] Alexa, "Top 1,000,000 sites," 2012. `http://www.alexa.com/topsites`.

[27] S. Hätönen, A. Nyrhinen, L. Eggert, S. Strowes, P. Sarolahti, and M. Kojo, "An experimental study of home gateway characteristics," in *Proceedings of the 10th annual conference on Internet measurement*, IMC '10, (New York, NY, USA), pp. 260–266, ACM, 2010.

[28] M. Zalewski, "Passive OS fingerprinting tool," 2003. http://lcamtuf.coredump.cx/p0f.shtml.

[29] A. Ramachandran and N. Feamster, "Understanding the network-level behavior of spammers," in *Proceedings of ACM SIGCOMM*, Sept. 2006.

[30] J. Demšar, B. Zupan, G. Leban, and T. Curk, "Orange: From experimental machine learning to interactive data mining," in *Knowledge Discovery in Databases: PKDD 2004*, vol. 3202, pp. 537–539, 2004.

[31] K. Kira and L. A. Rendell, "A practical approach to feature selection," in *Proceedings of the ninth international workshop on Machine learning*, ML92, (San Francisco, CA, USA), pp. 249–256, Morgan Kaufmann Publishers Inc., 1992.

[32] S. Hao, N. A. Syed, N. Feamster, A. G. Gray, and S. Krasser, "Detecting spammers with snare: spatio-temporal network-level automatic reputation engine," in *Proceedings of the 18th conference on USENIX security symposium*, 2009.

[33] X. Hu, M. Knysz, and K. Shin, "Measurement and analysis of global ip-usage patterns of fast-flux botnets," in *INFOCOM, 2011 Proceedings IEEE*, pp. 2633 –2641, Apr. 2011.

[34] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1." RFC 2616 (Draft Standard), June 1999. Updated by RFCs 2817, 5785.

[35] Maxmind, "Maxmind ip geolocation," 2012. `http://www.maxmind.com`.

[36] Symantec, "Tunable network parameters on linux," 2011. `http://www.symantec.com/business/support/index?page=content&id=HOWTO64299`.

[37] K. Kira and L. A. Rendell, "The feature selection problem: Traditional methods and a new algorithm," in *AAAI-92 Proceedings*, (Washington, DC, USA), Association for the Advancement of Artifical Intelligence, 1992.

[38] K. Thompson, G. Miler, and R. Wilder, "Wide-area internet traffic patterns and characteristics," *IEEE Network*, vol. 11, pp. 10–23, 1997.

[39] D. Miller, "Softflowd: A software netflow probe," 2012. `http://www.mindrot.org/projects/softflowd/`.

[40] M. Fullmer and S. Romig, "The OSU Flow-tools Package and Cisco NetFLow Logs," in *Proceedings of the 14th USENIX Large Installation Systems Administration Conference (LISA)*, Dec. 2000.

THIS PAGE INTENTIONALLY LEFT BLANK

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Marine Corps Representative
   Naval Postgraduate School
   Monterey, California

4. Directory, Training and Education, MCCDC, Code C46
   Quantico, Virginia

5. Director, Marine Corps Research Center, MCCDC, Code C40RC
   Quantico, Virginia

6. Marine Corps Tactical Systems Support Activity (Attn: Operations Officer)
   Camp Pendleton, California

7. Dr. Robert Beverly
   Naval Postgraduate School
   Monterey, California

8. Dr. Joel Young
   Naval Postgraduate School
   Monterey, California