2012-12

Transport traffic analysis for abusive infrastructure characterization

Nolan, Le; Beverly, Robert; Xie, Geoffrey

Monterey, California : Naval Postgraduate School

NPS-CS-12-005

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

TRANSPORT TRAFFIC ANALYSIS FOR ABUSIVE INFRASTRUCTURE CHARACTERIZATION

by

Le Nolan
Robert Beverly
Geoffrey Xie

December 14, 2012

THIS PAGE INTENTIONALLY LEFT BLANK

NAVAL POSTGRADUATE SCHOOL
Monterey, California 93943-5000

RDML Jan E. Tighe
Interim President

O. Douglas Moses
Acting Provost

The report entitled *"Transport Traffic Analysis for Abusive Infrastructure Characterization"* was prepared for and funded by National Science Foundation.

Further distribution of all or part of this report is authorized.

This report was prepared by:

Le Nolan
Department of Computer Science

Robert Beverly
Department of Computer Science

Geoffrey Xie
Department of Computer Science

Reviewed by:

Peter Denning, Chairman
Chairman
Department of Computer Science

Released by:

Jeffrey D. Paduan
Vice President and
Dean of Research

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)*
14–12–2012	Technical Report	2011-06-01—2012-06-31

4. TITLE AND SUBTITLE

Transport Traffic Analysis for Abusive Infrastructure Characterization

5a. CONTRACT NUMBER

5b. GRANT NUMBER

OCI-1127506

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Le Nolan, Robert Beverly, Geoffrey Xie

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

NPS-CS-12-005

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

National Science Foundation
Arlington, VA 22230

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 report 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

We investigate a promising approach that identifies discriminating features of likely communications involving abusive hosts from per-packet TCP header and timing information. These features identify congestion, flow-control, and other low-level network and system characteristics indicative of an abusive network host. Our approach is IP address and content agnostic, and therefore privacy-preserving to permit wider deployment than previously possible. Importantly, the modeled 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. In contrast to existing network-centric approaches reliant on flow-level records, fine-grained per-packet features yield superior performance with negligible processing impact. On real-world traces from accessing ∼40,000 Alexa and ∼30,000 known-abusive web sites, we achieve a classification accuracy of ∼94% with a 3% false positive rate using only transport features.

15. SUBJECT TERMS

Abusive Traffic, Traffic Analysis

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Robert Beverly
Unclassified	Unclassified	Unclassified	UU	43	19b. TELEPHONE NUMBER *(include area code)* (831) 656-2132

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

THIS PAGE INTENTIONALLY LEFT BLANK

Abstract

We investigate a promising approach that identifies discriminating features of likely communications involving abusive hosts from per-packet TCP header and timing information. These features identify congestion, flow-control, and other low-level network and system characteristics indicative of an abusive network host. Our approach is IP address and content agnostic, and therefore privacy-preserving to permit wider deployment than previously possible. Importantly, the modeled 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. In contrast to existing network-centric approaches reliant on flow-level records, fine-grained per-packet features yield superior performance with negligible processing impact. On real-world traces from accessing $\sim$40,000 Alexa and $\sim$30,000 known-abusive web sites, we achieve a classification accuracy of $\sim$94% with a 3% false positive rate using only transport features.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1: Introduction

Abusive traffic abounds on the Internet, in the form of email, malware, vulnerability scanners, worms, denial-of-service, drive-by-downloads, scam hosting, CAPTCHA solvers, and other, yet-to-be imagined 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. Contemporary techniques for obtaining and utilizing such infrastructure for abusive purposes are increasingly sophisticated and economically driven.

A large body of prior work 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 e.g., distributing signature databases, maintaining IP reputation, etc. Yet, despite years of commercial and research efforts, abusive traffic continues to impart both direct and indirect damage upon users, service providers, and the Internet.

Our research investigates a new passive traffic analysis approach to detecting abusive infrastructure that *does not rely on content inspection, sender reputation, or communication patterns* – features that are brittle and readily evaded. Instead, our approach utilizes *transport-layer* traffic analysis, a technique that has shown promise in the domain of fighting abusive email [6, 18, 24].

In contrast to previous efforts [16] that leverage aggregate traffic flow information such as that embedded in Netflow [7] records, we discover that fine-grain properties of the transport-layer packet stream, e.g., TCP retransmissions, packet reordering, arrival jitter, congestion and flow-control behavior, etc., are a *fundamental* source of discriminative information that can reliably identify 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 the 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, etc) and host (e.g., older computers and operating systems, high utilization, etc) characteritics. This basic weakness manifests in the resulting traffic stream, providing 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 previous ignored and logically orthogonal features of the network traffic, it can be usefully combined with existing classification and detection mechanisms to boost their performance or reduce load. For instance, the transport traffic technique

can be employed as an early reject before performing more costly deep packet inspection.

In addition to providing a new method to detect abusive traffic, our approach provides important 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.

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.

The remainder of this paper is organized as follows. Section 2 reviews prior work, while §3 details our methodology and data collection. In §4, we demonstrate the real-world efficacy of transport traffic analysis. We conclude in §6 with a summary of the larger implications of our work as well as suggestions for future research.

CHAPTER 2:
Related Work

The increasing variety of scams and attacks generating abusive traffic is resulting in a commensurate amount of research in understanding and defending against such traffic. Several works confirm that while there may be a larger number of nodes participating in the scam, much of their functionality is simply obfuscating the true origin of the scam content. Anderson *et al.* first demonstrated that a large number of nodes simply redirect or proxy traffic to a much smaller set of hosts. Their SpamScatter [5] tool follows URLs within spam messages to their final landing page where the page is rendered for image shingling. Based on image similarity metrics, SpamScatter finds approximately 2,000 scams on 7,000 servers.

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.

In a similar spirit, Levchenko *et al.* find that the large number of nodes that funnel traffic to a smaller set of nodes hosting content are in fact part of an even smaller set of scam campaigns and organizations [20]. In particular, the weak point in the scam chain is monetizing clicks which depends largely on foreign credit card processing. While this recent work reveals additional important properties of scams, they focus on scams that involve selling products. Our traffic based approach is more general, applicable to a variety of scams, malware, phishing, and other types of abusive traffic including active attacks.

Other approaches rely entirely on unique properties of the URL itself, for instance detecting that the URL was machine generated [21]. RBSeeker [16], on the other hand, differentiates malicious sites as those that induce an abnormally large number of redirections, i.e., HTTP or JavaScript. Unfortunately, we discover in §4.1 that redirections are common on legitimate web sites, either as part of the Content Distribution Network (CDN) functionality, or to customize content on per-device basis. For instance, web sites often recognize the user-agent of mobile devices and perform legitimate redirection as a result. While whitelists can be employed to avoid false-positives due to legitimate web sites, such lists are brittle and must be continually maintained.

A closely related focus is finding "botnets." Botnets are collections of compromised hosts operating under common control and are particularly well-suited to provide a distributed computing platform for nefarious purposes. Existing botnet defenses have been unsuccessful largely because the operators of botnets have a strong incentive to quickly adapt and evade defense mechanisms. For instance, earlier approaches attempt to discover the centralized command and control channel of botnets [12]. This traditional Achilles heel of botnets is no longer an effective defense as modern botnets use distributed peer-to-peer control.

Other promising work unfortunately relies on brittle heuristics and unreliable identifiers. For instance, Xie *et al.* provide a system [28] to identify and characterize botnets using an automatic technique based on discerning spam URLs in email. In contrast, our approach is content agnostic. Other research relies on IP addresses as identifiers [30]. Yet, botnet IP addresses are highly dynamic, drawing upon a large pool of fresh addresses . Similarly, DNS is a poor identifier given the prevalence of botnet fast-flux [15] to rapidly change the DNS entries of scam hosting infrastructure. In sum, IP and content-based techniques can, and will, be subverted by motivated attackers.

Zhao *et al.* tackle the difficult problem of identifying bots that are used to signup for free web mail from major providers as part of a scam campaign [30]. Because an individual account may only send a few messages, from a well-known and trusted provider, they attempt to evade detection. The BotGraph work attempts to detect such accounts by constructing large user-user graphs and looking for tightly connected subgraph components.

Botminer an application created by Gu *et al.* captures communication structure from network traffic [11] . Combined with IDS-like functionality at higher layers, they cluster hosts with similar flows where the intuition is that bot hosts will have different communication patterns than legitimate hosts. While our goal is similar to BotMiner, our approach is substantially different. Rather than looking at the communication structure, we focus on properties of the traffic stream (and we ignore source and destination IP addresses entirely). Further, we examine none of the application-layer content. As alluded to previously, our technique could be combined with BotMiner-like functionality where permitted by privacy constraints, or used in isolation when required.

Finally, most closely related to the presented research, is the successful application of transport traffic analysis to the spam reception problem. Rather than relying on IP reputation or content inspection, statistical traffic signal characterization of the type we employ here can differentiate between valid and spam messages on the Message Transfer Aget (MTA) with high accuracy [6, 18, 24]. Whereas this technique has been applied to the reception of abusive messages, we investigate the logical dual: whether traffic features can identify hosts serving abusive content. As we will demonstrate, the technique generalizes well, but hones in on substantially different features from those of spam identification.

CHAPTER 3: Methodology

This section describes our experimental methodology, including our sample host populations, transport traffic classification procedure, and collection of a real-world dataset on which to evaluate our technique.

3.1 Host Populations

We gather four populations of legitimate and abusive hosts of various types. First, we manage a spam honeypot where no valid accounts exist, thus, only messages known by definition to be abusive arrive. Each message is processed to extract all embedded URLs. Spam messages frequently contain URLs from legitimate web sites, e.g., they make use of hypertext references to external images, style sheets, etc., residing on a legitimate company's website in order to reduce the spammer's bandwidth costs or make the email appear more legitimate. We therefore filter the URLs extracted from our honeypot against a blocklist before insertion into the database.

To obtain a wide variety of URLs for broad analysis, we include known malware domains from two sources ([1], [2]) as well as known phishing sites collected by [3]. For legitimate URLs, we use the Alexa [4] database. We make use of the top 20,000 most popular websites as ranked by Alexa, as well as a random sampling of 20,000 sites from the complete Alexa one million list. While we collect over 1.4M URLs from our honeypot, we reduce these to 9,678 fully unique URLs by removing duplicates, duplicated domains, and whitelisted URLs.

Many of the same URLs are in the different external malware and phishing lists. The union of the two malware lists and phishing list results in 20,013 unique URLs that we term our "external" dataset. However, the relative unreliability of the honeypot and external abusive hosts causes our fetcher, described next, to successfully download the content of only approximately 50-70% of URLs, as summarized in the yield column of Table 3.1. The final column in Table 3.1 shows the total number of fetches per population when including redirection chains.

Table 3.1: 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	6.8K	9.4K
Total Fetches	27.9K	25.3K	7.9K	12.5K

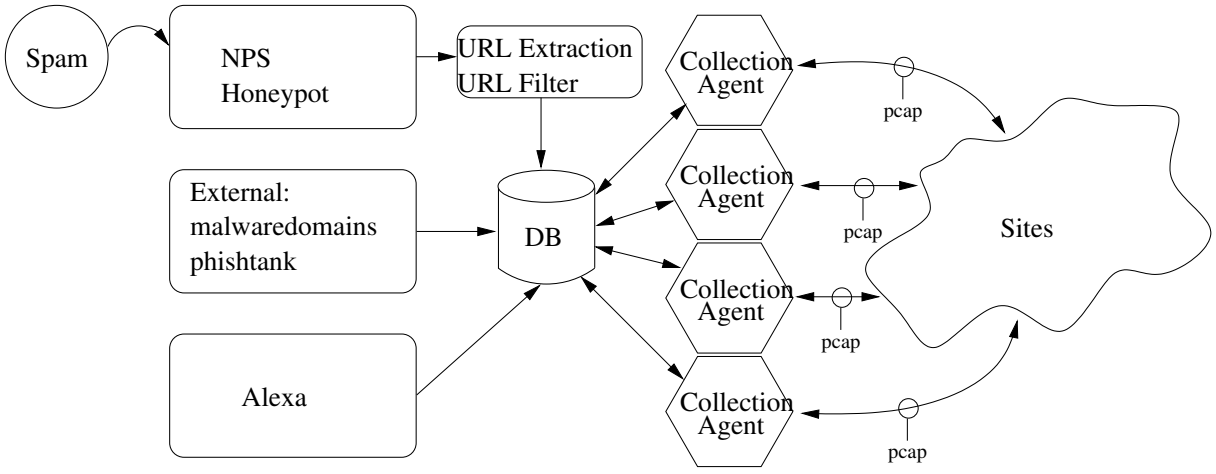


Figure 3.1: Data Collection: Multiple sources of abusive and legitimate URLs are probed by collection agents (Table 3.2) with varied connectivity. Contemporaneously, collection agents capture packets.

3.2 Data Collection

Figure 3.1 illustrates our data collection process. We implement a custom URL content fetcher and follower. To measure redirection, we follow HTTP "content moved" status codes, HTML meta-refresh, frames, and non-obfuscated JavaScript [9]. The fetcher respects cookies and supports TLS when required, allowing us to successfully follow redirection chains where abusive hosts attempt to employ these for protection. Our JavaScript redirection covers traditional calls to `window.location`, `location.href`, `location.replace`, `window.navigation` `self.location`, and `top.location`. Where JavaScript was intentionally obfuscated, we do not follow any redirection. We discuss the prevalence of obfuscation in §3.3.

We follow all redirection chains to their end, wherever possible. For each HTTP object, we record the ephemeral source TCP port, a timestamp, the size of the object retrieved, the parent URL, and the current point along any redirection chain. Section 4.1 examines the types and prevalence of redirection behavior we observe in detail.

From January 25 to 27, 2012, we deployed our fetcher on four distinct collection agents, running FreeBSD and Linux TCP stacks, which we term "vantage points." As detailed in Table 3.2, each agent has different network connectivity and physical attachment points. Because we examine detailed characteristics of the network traffic stream, including congestion and delay effects, it is important to adequately represent a variety of potential operating environments. Our collection agents therefore span three orders of bandwidth magnitude, different multiplexing, are located on both east and west coasts of the United States, and 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 §4.4.

During collection, we capture all packets as pcap-formatted traces. Packets are coalesced into bidirectional flows according to distinct tuples of: destination IP address, and ephemeral source

Table 3.2: 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

TCP port. Note that this flow definition differs from that of e.g., Netflow [7] as our collection agent's IP addresses and the remote TCP port are fixed. As flows complete, either via an explicit TCP termination or via a 300s timeout, we extract transport-layer features. By recording the ephemeral port during the content fetch, we match flows to their respective content.

3.3 Fetcher Validation

While following each redirection chain, we find and record instances where exceptions occur. Recording exceptions allows us to understand how often, and in what way, our fetcher fails to pull content. The most common exceptions involve the initial TCP three-way handshake with the host corresponding to the URL being fetched. These include timeouts, refused connections, connection resets, and DNS failures. These types of error on the initial connection account for 90%, 94%, 93%, and 99% of the total exceptions, in the Alexa, Alexa random, honeypot, and external datasets respectively.

Because most exceptions we encounter are the result of connection failure, Table 3.3 provides columns for both all exceptions, as well as "non-connect:" exceptions caused by a fault other than a failed connection. Examples of non-connect exceptions include bad HTTP status codes, missing or blank redirection locations, malformed redirects, etc. Table 3.3 further divides the frequency of 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 750 exceptions among the Alexa top 20,000 sites. These include 357 timeouts, 327 DNS failures, 16 connection resets, 20 connections refused, and 11 route failures. The remaining 20 include redirection errors. 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 would be 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 honeypot URLs where we see under 3%. Thus, the majority of failures are decoupled from our fetching procedure.

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 3.1 (1000 random samples in

Table 3.3: 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%

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 URL and content of the final page in the redirection chain of both the fetcher engine and web browser. In all cases, we accurately capture instances of no redirection and redirection due to HTTP status codes or HTTP headers.

The most prevalent case of inconsistency is due to obfuscated JavaScript. Within the population of honeypot and external URLs, we randomly sample 350 sites with JavaScript for manual inspection. Of these, 20 (6%) contained obfuscated JavaScript using common techniques such as hex values, splitting text with variables, etc. In general, however, the JavaScript in the abusive sites was either clearly redirection or used for other functions.

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 otherwise achieve their objective. The resulting abusive infrastructure is therefore frequently either compromised hosts, for instance, rent-per-hour botnets [27], or complicit service providers. In either case, these hosts are typically poorly connected, under-provisioned, and overloaded (in terms of either communication or computation).

For example, botnet hosts can be home user's computers that are unwittingly compromised, i.e., residential machines without enterprise-level infrastructure. These botnet hosts are thus resource constrained, both in terms of available computational power and bandwidth. Moreover, these residential Internet connections are typically links with asymmetric bandwidths, e.g., aDSL and cable modems – links that impart distinct signatures in their traffic stream.

Similarly, complicit service providers willing to support abusive infrastructure are frequently located in countries and companies with poor Internet connectivity, low bandwidth, limited peering, etc.

Exacerbating cases of poor connectivity is 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. As malicious traffic congests available capacity, local queues form on the host operating system and residential gateway. In fact, the local buffer sizes on residential cable and DSL modems is quite large [14]. Thus, it is reasonable to expect TCP timeouts, retransmissions, resets, out-of-order packets, highly variable round trip time (RTT) estimates, etc[1]. This traffic signature is distinguishable using statistical learning techniques. Table 3.4 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 have only 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 [29]. 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 difficult for abusive hosts to subvert.

We therefore do not attempt to make use of reputation measures such as IP addresses, or perform inference over IP addresses. Further, our technique does not require deep-packet inspection, a costly measure that can be unreliable. 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. A fast discriminator for in-core use, or 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 3.4. We use a variety of well-known supervised statistical learning algorithms, including Naïve Bayes, Decision Trees, and Support Vector Machines (SVM), as

[1] See [25] for a description of RSTs, FINs, congestion windows and other details of the TCP specification.

Table 3.4: 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

implemented in third-party software [8]. The details of each of these algorithms is outside the scope of this work; we employ them as tools and to evaluate their relative performance.

Specific to our testing, the SVM uses the C support vector classification algorithm, where $C = 1$ is the error minimization cost factor, with a radial basis kernel where $\gamma = 1/N$ for N training samples. The Naïve Bayes algorithm uses the m-estimate instead of relative frequency when computing the class priors. Finally, the decision tree uses the information gain ratio to determine tree splits.

Importantly, we employ 10-fold cross-validation in all experiments herein. 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. 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, we wish to understand which feature is providing the most discriminative power, i.e., what properties of the traffic stream are predictive. Some learning algorithms, such as decision trees, naturally provide the best features as part of their output. Others, such as SVMs, require feature selection methods. In this paper, we make use of a variant of the Relief algorithm [19] for feature selection that is included in our third-party software [8].

Performance Metrics

We use standard classification performance metrics. We canonically 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:

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

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

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

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

$$F-score = 2\left(\frac{precision*recall}{precision+recall}\right) \tag{3.5}$$

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 4:
Transport Traffic Analysis

In this section, we first discuss the relation of our approach to common Netflow records in general, and schemes that rely on redirection behavior evident in Netflow in particular. We examine the types and frequency of redirection we discover on our datasets, as well as different feature distributions. Next, we apply our classification scheme to HTTP traffic gathered from known good and abusive hosts across four different vantage points. In addition to the classifier performance, we discuss the the most discriminative features, and the sensitivity of our model to legitimate sources of congestion.

4.1 Netflow Limitations

Significant prior work has explored various aspects of network-level information to infer various nefarious activity. We stress here that our approach is distinct from current practice: rather than examining the distribution of abusive IP addresses [13], network BGP prefixes [26], IRC channel [12] or peer-to-peer botnet signatures [11], DNS behavior [15], or web page image shingling [5], our approach is content, name, and IP address agnostic.

Most closely related to our effort is work that uses Netflow [7] 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 flow record contains coarse-grained information such as total number of bytes, packets, and flow duration. Thus, Netflow contains only a small subset of the features in Table 3.4.

One example of using Netflow records for abusive botnet host detection comes from "redirection bot seeker," [16] or RBseeker[1]. In RBseeker, Netflow records are used to identify infrastructure bots that use redirection to obfuscate and hide the true source of abusive content.

Per-device Redirection is Common

However, an initial investigation of the Alexa top 20,000 sites suggests that redirection is becoming common. One root cause of legitimate redirection is websites that use 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 new paper will commonly be redirected to a mobile version of the content suitable to the device's screen, network speed, etc.

To understand redirection due to the end-user's device, we modify our HTTP poller to present 12 different "User-agent" strings to each of the Alexa web servers. As part of the exchanged HTTP headers, the "User-agent" identifies the software and hardware of the requesting browser [9].

[1]Other schemes employ Netflow; our intent is to identify instances where fine-grained packet features are superior to Netflow analysis.

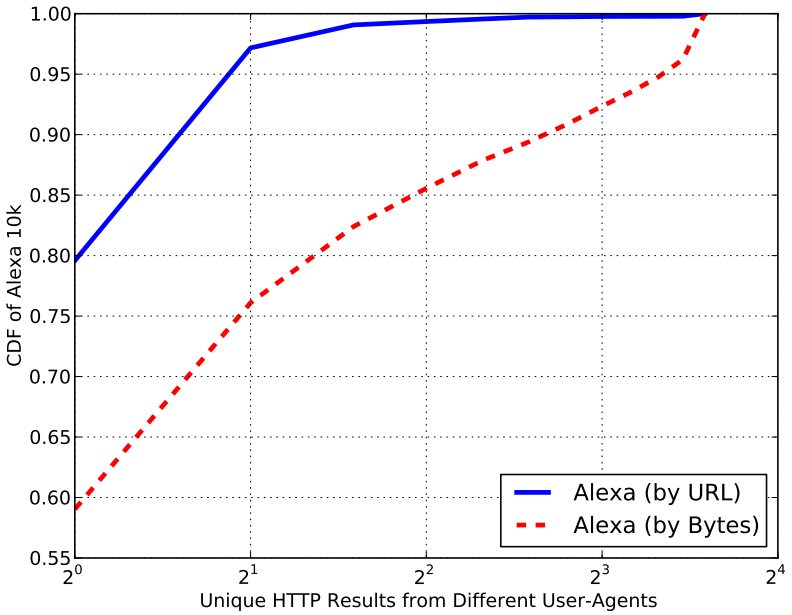


Figure 4.1: Redirection by per-device customization.

We test User-agent strings representing all major browsers, operating systems, and a variety of mobile devices.

For each 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 redirects users when they report to be e.g., using a mobile device will increase the set size.

Figure 4.1 demonstrates that approximately 20% of the Alexa 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.

General Redirection is Common

User-agent based redirection is just one source of legitimate redirection. Next, we wish to understand the types and prevalence of various redirection we encounter when visiting both legitimate and known abusive web sites in Table 3.1. Note, in this analysis, our fetcher presents only a single HTTP User-agent string to each web server.

Figure 4.2 shows the fraction of sites that perform any type of redirection for our various datasets, as a function of vantage point. As expected, there is little difference in behavior attributable to vantage point, with the exception of malware and honeypot sites where site avail-

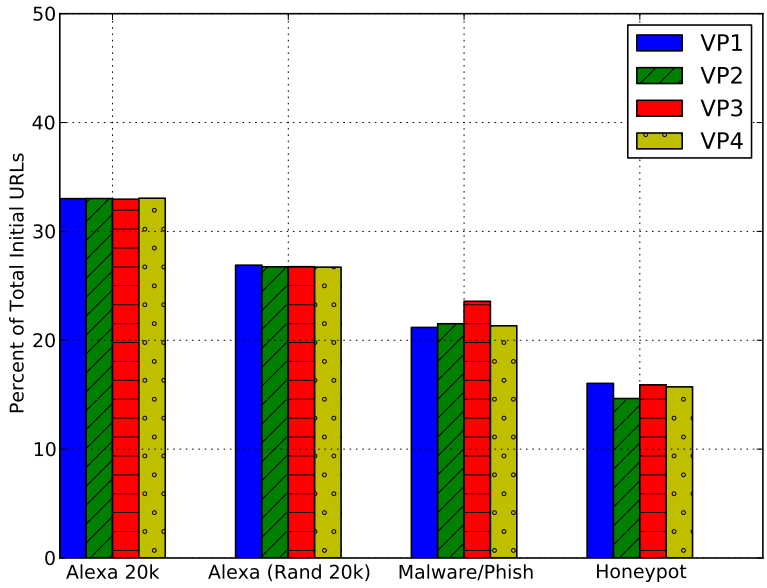


Figure 4.2: Frequency of redirection among web site populations, as observed from four vantage points.

ability can vary instantaneously.

Surprisingly, the Alexa top 20K sites perform the most redirection, while the malware and honeypot sites only redirect approximately 20% of the time. Thus, even if redirection were a sufficient means by which to detect abusive infrastructure, abusive sites have compensated and use different obfuscation techniques, e.g., proxying and DNS redirection. Note that 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. 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.

Irrespective of legitimate or abusive, when a site redirects, it overwhelmingly does so via HTTP `3xx` status codes [9]. 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 a similar redirection character as a random selection of 20,000 Alexa sites.

A negligible fraction of sites uses 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 redirection. Meta refresh is a tag that is part of the HTML web page content rather than the

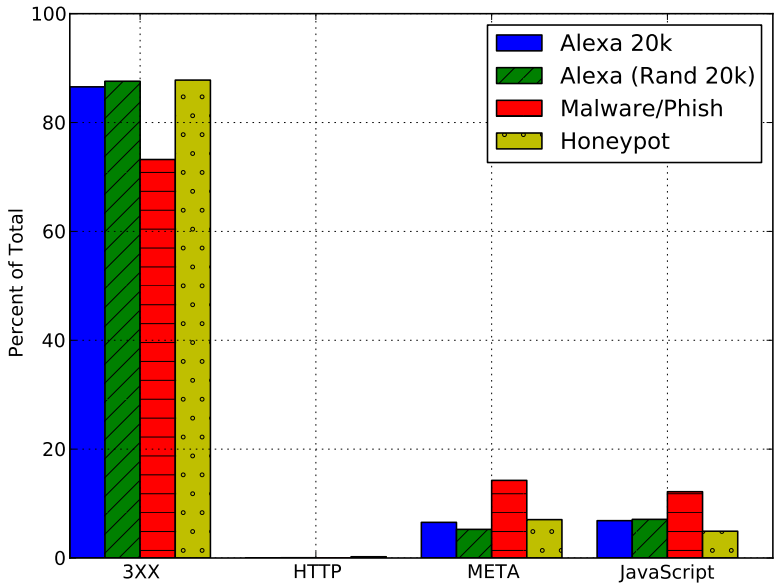


Figure 4.3: Redirection technique employed when a web site redirects.

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.

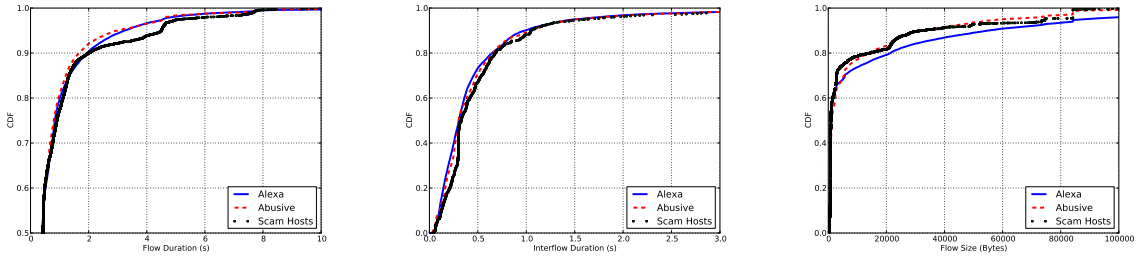
Netflow is too coarse-grained

The three primary features in e.g., [16] 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), that 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 in this role, we examine the distribution of these features in 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 3.1, 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.

Figure 4.4 plots the cumulative fraction of each of these Netflow features as observed in our three datasets: the Alexa hosts, external abusive hosts, and our honeypot-gathered scam hosts.



(a) Duration (b) Interflow Duration (c) Bytes

Figure 4.4: Distribution of features available from Netflow evident in our datasets.

We see that the total time duration (4.4(a)) and interflow durations (4.4(b)) are largely indistinguishable between the legitimate and abusive sites. While the total number of bytes of Alexa hosts is generally larger and has a longer tail (4.4(c)), the results are again quite similar.

Redirection is only one potential means that abusive infrastructure utilizes to remain hidden and protected; proxies are a second. In addition to the relative lack of differentiation afforded by redirection, such techniques cannot detect proxies.

Thus, our fundamental contention is that 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, we perform basic data exploration to understand which features are likely to be discriminative, as well as to substantiate or disprove our intuitions over abusive host traffic. We also wish 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," corresponding to the first URL, the set of any intermediate URLs due to redirection, and the final landing page due to any redirection.

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 [22]. Table 4.1 shows two interesting facts: first, continental host distribution varies little among abusive and legitimate sites, suggesting that location or RTT-based techniques cannot suffice to identify abusive hosts. Second, North America dominates across all populations, with

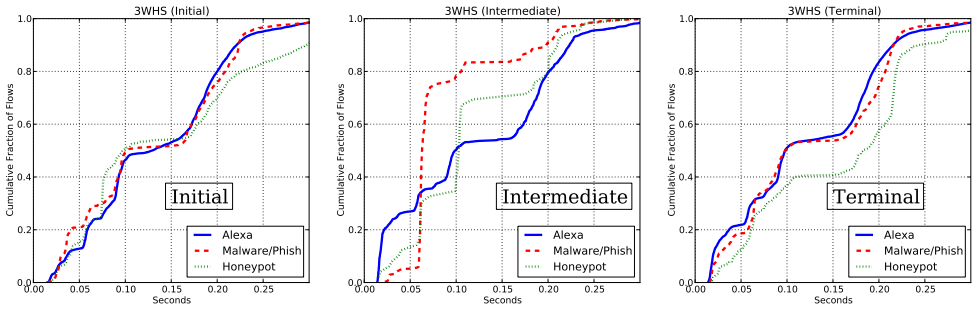


Figure 4.5: Distribution of initial RTT from TCP three way handshake among different website populations.

Table 4.1: 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

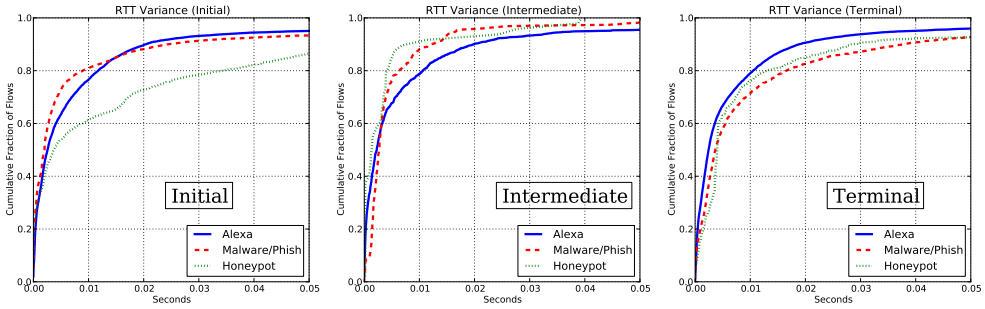


Figure 4.6: Distribution of RTT variance among different website populations.

between 50-52% of the hosts.

We therefore next turn to the variance of RTT samples, as inferred by our technique. Figure 4.6 shows that RTT variance is a strong indicator for abusive hosts of scam infrastructure as discovered by our honeypot, especially for the initial and terminal URL. Whereas the variance is consistent across the legitimate redirection chain, we observe different characteristics for abusive hosts.

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

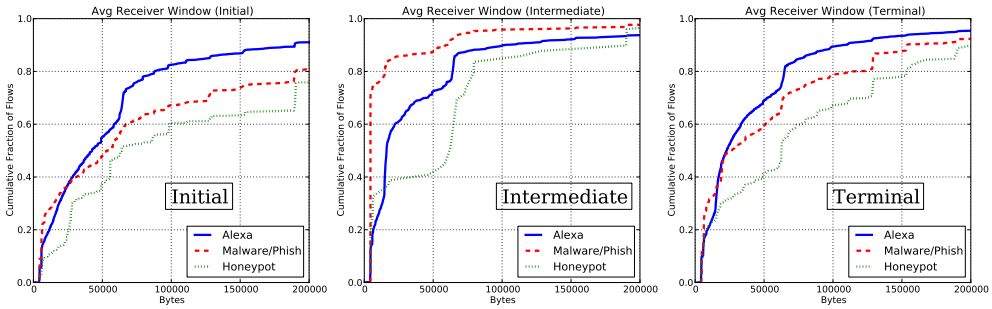


Figure 4.7: Distributions of average TCP receiver window size among different website populations.

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 [17]. 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. A detailed exploration of this phenomenon reveals that it is due mainly in part of 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 that 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). 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 the three binary classification, supervised learning algorithms across different combinations of legitimate and abusive hosts in our population (Table 3.1), for traffic as observed at our four different vantage points (Table 3.2).

In assessing classification performance, all of the metrics in §3.5 are important to consider. For instance, accuracy is misleading if the underlying class prior is heavily skewed, e.g., if there are many more legitimate than abusive hosts being classified. Precision therefore measures, for hosts predicted as abusive, the fraction that are truly abusive, as determined by the population

Table 4.2: Transport traffic classification performance

	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 4.3: 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

from which they originated. Recall measures the influence of misclassified hosts, i.e., 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 that the decision tree learner performs better than either the Naïve Bayes or SVM. We therefore present only results from the decision tree in the remainder of the results. Table 4.2 shows our performance results for different combinations of legitimate and abusive sites using decision trees for VP4. Our first observation is that the transport features can differentiate the set of 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, the external sites give an 87% accuracy.

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.

Next, we examine the classification F-scores for each of our four vantage points. 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 4.3 shows that 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 that the technique works well for a variety of different connection types and physical locations.

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 that transport traffic analysis can greatly aid existing schemes by providing a method that 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.

4.4 Congestion Sensitivity

A natural concern with our approach is that the traffic features we extract are often indicative of congestion, suggesting that legitimate hosts experiencing congestion might be misclassified as abusive. To better understand the sensitivity of our technique, we analyze the traffic of legitimate hosts that are congested with 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.

We setup 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.

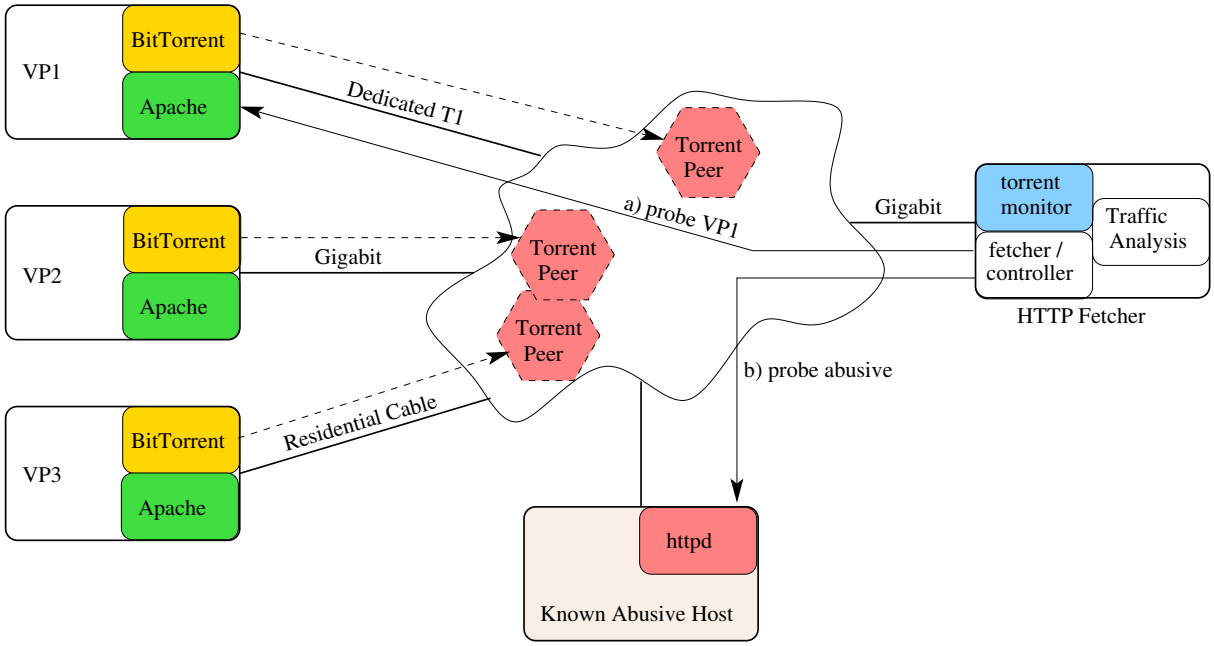
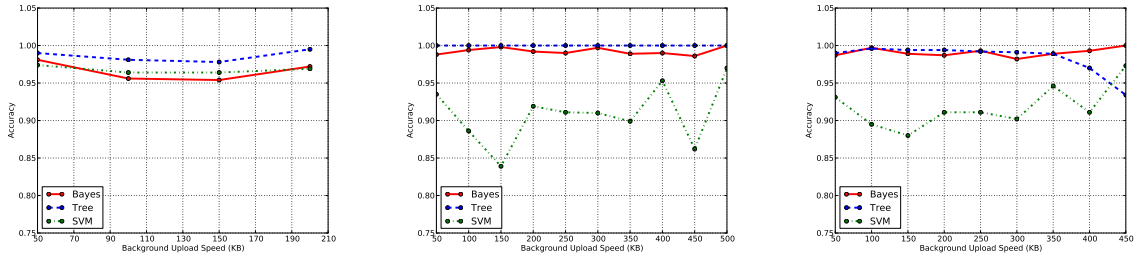


Figure 4.8: Test and Setup

The BitTorrent client seeds many public, legal Torrents. Recall that each of the vantage points reside on networks with varying access connectivity, so that 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 resulting from collection.



(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 4.9 depicts the transport traffic analysis performance, measured by F-score, as a function of background congestion and link connectivity. Note that 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.

Perhaps counter-intuitively, our classification performance remains high in the face of background traffic. More surprisingly, the performance appears to be insensitive to the level of background traffic. Across all background traffic levels, we continue to see that the receiver window features are most discriminatory as these are insensitive to background cross-traffic. However, we find that 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.

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 that 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. Again, we see the set of 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.

4.5 Overhead

Finally, we examine the performance impact of our transport traffic collection agent, as compared to industry-standard Netflow. To mimic the overhead of Netflow, we employ two open-source tools: flowtools as a Netflow collection agent [10], and softflowd [23] to read a pcap capture file, create flows, and export them to the flowtool collector.

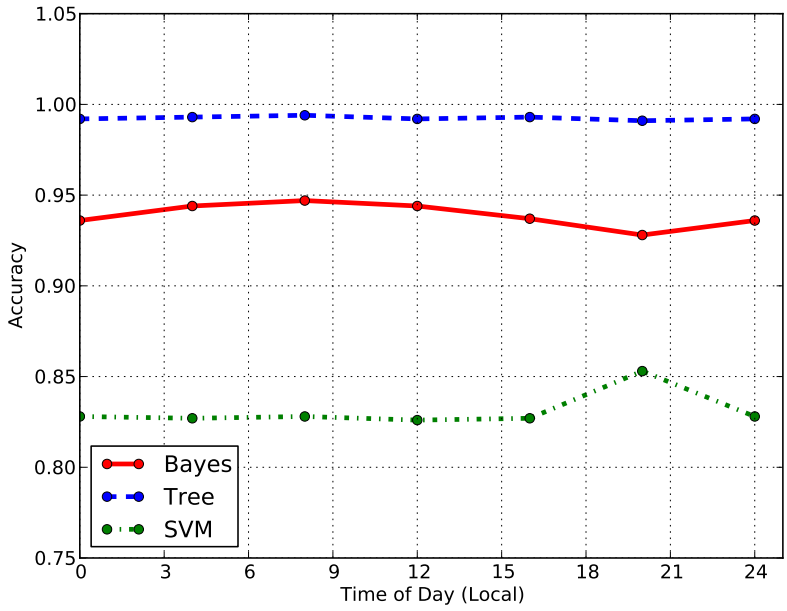


Figure 4.10: Transport traffic analysis performance as a function of time-of-day

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 55 bytes transforming in 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.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 5:
Discussion

The transport traffic features we explore are low-level: some are inherent to the environment, e.g., cross-traffic, packet reordering, timing, etc, whereas other are specific to the end-system, e.g., receiver window behavior, congestion etc. These features are therefore impossible or difficult to modify without detection.

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 that such efforts will be unreliable. Other aspects, such as the receiver window behavior require operating system and kernel modifications that are hard to effect. "Low-and-slow" type of stealth strategies [16] may be effective in evading our transport traffic technique, but impart a signficant cost on abusive infrastructure as it must be more widely distributed to achieve reasonable aggregate rates.

A natural question is whether the 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," etc.

Surprisingly, only 2% of the unique IPs are inferred as residential based on the DNS name heuristics. Another approximately 37% have no DNS PTR record. The remaining 61% are inferred as 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 imbalance in data complexion. The F-score is poor, at only 9%. We leave the investigation of finding bots among the abusive hosts for future work.

Finally, while this paper does not focus on network congestion, we recognize that the 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 would 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. We plan to perform further experiments to better understand this phenomenon.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 6:
Conclusions

This paper explores a new passive traffic analysis approach to detecting abusive infrastructure that does not rely on content inspection, sender reputation, or communication papers. Rather, we show a promising approach based on fine-grained *transport traffic features*, as opposed to existing Netflow analysis methods.

Our results demonstrate that per-packet TCP header and timing information alone can detect characteristics of abusive traffic, and hence suggests at a variety of means to detect and block abusive infrastructure. This IP address and content agnostic approach can be more widely deployed than previously possible, for instance in networks or countries with strict privacy constraints.

In future work, we plan to understand the power of combining methods, for instance using transport features as part of a multi-stage hypothesis test. We note that 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. Finally, we hope to apply the transport traffic analysis method to detecting automated attacks against legitimate web sites, including CAPTCHA solvers, vulnerability scans, etc.

THIS PAGE INTENTIONALLY LEFT BLANK

REFERENCES

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

[2] Malware domain list, 2012. http://www.malwaredomainlist.com/.

[3] Phish tank, 2012. http://www.phishtank.com/.

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

[5] David S. Anderson, Chris Fleizach, Stefan Savage, and Geoffrey M. Voelker. Spamscatter: characterizing internet scam hosting infrastructure. In *Proceedings of 16th USENIX Security Symposium*, pages 10:1–10:14, 2007.

[6] Robert Beverly and Karen Sollins. Exploiting transport-level characteristics of spam. In *Proceedings of the Fifth Conference on Email and Anti-Spam (CEAS)*, August 2008.

[7] B. Claise. Cisco Systems NetFlow Services Export Version 9. RFC 3954 (Informational), October 2004.

[8] Janez Demšar, Blaž Zupan, Gregor Leban, and Tomaz Curk. Orange: From experimental machine learning to interactive data mining. In *Knowledge Discovery in Databases: PKDD 2004*, volume 3202, pages 537–539. 2004.

[9] 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.

[10] Mark Fullmer and Steve Romig. The OSU Flow-tools Package and Cisco NetFLow Logs. In *Proceedings of the 14th USENIX Large Installation Systems Administration Conference (LISA)*, December 2000.

[11] Guofei Gu, Roberto Perdisci, Junjie Zhang, and Wenke Lee. Botminer: Clustering analysis of network traffic for protocol- and structure-independent botnet detection. In *USENIX Security Symposium*, pages 139–154, 2008.

[12] Guofei Gu, Junjie Zhang, and Wenke 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.

[13] Shuang Hao, Nadeem Ahmed Syed, Nick Feamster, Alexander G. Gray, and Sven Krasser. Detecting spammers with snare: spatio-temporal network-level automatic reputation engine. In *Proceedings of the 18th conference on USENIX security symposium*, 2009.

[14] Seppo Hätönen, Aki Nyrhinen, Lars Eggert, Stephen Strowes, Pasi Sarolahti, and Markku Kojo. An experimental study of home gateway characteristics. In *Proceedings of the 10th annual conference on Internet measurement*, IMC '10, pages 260–266, 2010.

[15] Xin Hu, M. Knysz, and K.G. Shin. Measurement and analysis of global ip-usage patterns of fast-flux botnets. In *INFOCOM, 2011 Proceedings IEEE*, pages 2633 –2641, April 2011.

[16] Xin Hu, Matt Knysz, and Kang G. Shin. Rb-seeker: Auto-detection of redirection botnets. In *Proceedings of 16th Annual Network and Distributed System Security Symposium*, 2009.

[17] V. Jacobson, R. Braden, and D. Borman. TCP Extensions for High Performance. RFC 1323 (Proposed Standard), May 1992.

[18] Georgios Kakavelakis, Robert Beverly, and Joel 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)*, December 2011.

[19] Kenji Kira and Larry A. Rendell. A practical approach to feature selection. In *Proceedings of the ninth international workshop on Machine learning*, ML92, 1992.

[20] Kirill Levchenko, Andreas Pitsillidis, Neha Chachra, Brandon Enright, Márk Félegyházi, Chris Grier, Tristan Halvorson, Chris Kanich, Christian Kreibich, He Liu, Damon McCoy, Nicholas Weaver, Vern Paxson, Geoffrey M. Voelker, and Stefan 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, pages 431–446, 2011.

[21] Justin Ma, Lawrence K. Saul, Stefan Savage, and Geoffrey M. Voelker. Identifying suspicious urls: an application of large-scale online learning. In *ICML*, 2009.

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

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

[24] Tu Ouyang, Soumya Ray, Michael Rabinovich, and Mark Allman. Can network characteristics detect spam effectively in a stand-alone enterprise? In *Proceedings of the 12th Conference on Passive and Active Network Measurement*, March 2011.

[25] J. Postel. Transmission Control Protocol. RFC 793 (Standard), September 1981. Updated by RFC 3168.

[26] Anirudh Ramachandran and Nick Feamster. Understanding the network-level behavior of spammers. In *Proceedings of ACM SIGCOMM*, September 2006.

[27] Mathew Schwartz. Pssst...Want to Rent a Botnet?, 2010. http://www.darkreading.com/security/vulnerabilities/225200525.

[28] Yinglian Xie, Fang Yu, Kannan Achan, Rina Panigrahy, Geoff Hulten, and Ivan Osipkov. Spamming botnets: signatures and characteristics. *SIGCOMM Comput. Commun. Rev.*, 38(4):171–182, 2008.

[29] Michal Zalewski. Passive OS fingerprinting tool, 2003. http://lcamtuf.coredump.cx/p0f.shtml.

[30] Yao Zhao, Yinglian Xie, Fang Yu, Qifa Ke, Yuan Yu, Yan Chen, and Eliot Gillum. Botgraph: large scale spamming botnet detection. In *NSDI'09: Proceedings of the 6th USENIX symposium on Networked systems design and implementation*, pages 321–334, 2009.

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. Research Sponsored Programs Office, Code 41
   Naval Postgraduate School
   Monterey, CA 93943

4. National Science Foundation
   Arlington, VA 22230