RB-Seeker: Auto-detection of Redirection Botnets*

Xin Hu, Matthew Knysz, and Kang G. Shin
University of Michigan Ann Arbor
{huxin, mknysz, kgshin}@eecs.umich.edu

Abstract

*A **Redirection Botnet** (RBnet) is a vast collection of compromised computers (called bots) used as a redirection/proxy infrastructure and under the control of a botmaster. We present the design, implementation and evaluation of a system called **Redirection Botnet Seeker** (**RB-Seeker**) for automatic detection of RBnets by utilizing three cooperating subsystems. Two of the subsystems are used to generate a database of domains participating in redirection: one detects redirection bots by following links embedded in spam emails, and the other detects redirection behavior based on network traces at a large university edge router using sequential hypothesis testing. The database of redirection domains generated by these two subsystems is fed into the final subsystem, which then performs DNS query probing on the domains over time. Based on certain behavioral attributes extracted from the DNS queries, the final subsystem makes use of a 2-tier detection strategy utilizing hyperplane decision functions. This allows it to quickly identify aggressive RBnets with a low false-positive rate ($<$ 0.008%), while also accurately detecting stealthy RBnets (i.e., those mimicking valid DNS behavior, such as CDNs) by monitoring their behavior over time. Using DNS behavior as a means of detecting RBnets, RB-Seeker is impervious to the botmaster's choice of Command-and-Control (C&C) channel (i.e., how the botmaster communicates and controls the bots) or use of encryption.*

1 Introduction

Recently, botnets—a vast collection of compromised computers called *bots* under a common Command-and-Control (C&C) infrastructure—have have become one of the biggest threats to the Internet community, due to their financial appeal and the widespread world-wide adoption of broadband Internet connections. The critical difference between botnets and other known malware is the use of Internet Relay Chat (IRC), or other protocols, as a flexible and extensible method for a Command and Control (C&C) channel, enabling the coordination of thousands of individual bots to launch larger-scale and more powerful attacks. Moreover, bot malware is becoming more modular in nature and nearly all current bots support binary updates, allowing a botmaster (i.e., a botnet's controller) to increase functionality and evade various detection strategies. The use of updatable malware combined with a C&C channel affords the botmasters a great deal of control over the compromised computers (i.e., bots), providing a wide range of nefarious services and activities which are sold for profits. For example, a single botnet could be used to send spam emails, steal confidential information, and mount Distributed Denial of Service (DDoS) attacks, depending on the current needs of the botmaster or his/her employers. Often, due to the ease of acquiring a modest botnet at a low cost, control of the entire botnet is sold rather than just the services it can provide.

While the criminal uses of botnets are numerous, the more popular/profitable ones include: redirection to malicious content (such as fraudulent websites used in spam/phishing campaigns), confidential information theft, sending spam/phishing emails, and DDoS attacks against servers or even the Internet infrastructure of a country [23]. A significant amount of work has already been done on the subject of detecting and mitigating spam emails and DDoS attacks. In this paper, we focus on detecting bots (or other compromised systems) used in redirection/proxy scams. We term these bots *Redirection Bots* (RBs) and call the botnets they compose *Redirection Botnets* (RBnets). Since botnets are the primary source of such redirection endeavors, detecting computers partaking in suspicious redirection can provide us a critical means of detecting these botnets. Furthermore, a botnet's versatility allows it to provide multiple criminal services, and hence, by detecting and mitigating RBnets, we can help deter the other malicious activities they may invoke.

Botnets are essentially an abundant source of disposable redirection servers/proxies, which serve as the front-end to malicious content hosted elsewhere—on anything from a powerful central server to another bot. Used as a misdirection mechanism for evading detection, RBnets are used in

*The work reported in this paper was supported in part by NSF under Grant CNS-0523932.

tandem with other criminal scams, constituting only a portion of the overall operation. For example, spam/phishing campaigns often utilize a RBnet for misdirection. They begin by using some spamming mechanism (e.g., a hijacked mail server and/or a botnet) to send seemingly interesting phishing emails. Within the phishing emails are innocuously disguised embedded links pointing to a RBnet. Once victims click the embedded links, they connect to the bots which then redirect them to—or serve as proxies for—the actual host of the nefarious content. While this single layer of redirection is the simplest case, it is common for criminals to employ multiple layers of redirection between the victim and the malicious content host. Botnets are an attractive redirection mechanism, because if one is discovered and blocked, there is an ample supply of other bots to take its place; and the blocked bot can still be used for other villainous purposes. The use of RBnets for spam/phishing campaigns is so successful at protecting the malicious-content hosts, that criminals are beginning to centralize their operations. Numerous bots act as forwarding servers for the same phishing/scam campaigns, redirecting users to the same final-destination servers (called *motherships*) which host the illegal content. This strategy grants criminals a high level of anonymity via redirection while enabling easy centralized management.

While the RBnets can be used to deliver malicious content to victims via either redirection or proxy, redirection offers several financial and performance advantages over proxy in terms of content availability, resource utilization, and ease of management. Because botnets are composed primarily of compromised home computers with unreliable connectivity, it is common for them to unexpectedly go offline. If a botmaster is using bots as proxies to deliver malicious content to victims, the bots must remain online during the entire session. When a proxy bot unexpectedly goes offline, the connection between the victim and the source of the malicious content is severed. Using bots for redirection, on the other hand, is more resilient to connection failure because individual bots within the RBnet need to maintain connectivity only long enough to redirect the victim. As a result, the use of redirection will be more effective in terms of content availability. Moreover, individual bots experience more resource strain when used as proxies. Each bot must maintain connections with victims while serving as a proxy to the actual content host. Since bots are often less powerful compromised home computers, this limits the number of victims that can be serviced by an individual bot. A botmaster can achieve better utilization of the botnet's resources by using redirection; it is considerably less taxing on the compromised home computers composing the botnet. This greatly improves the financial gain achievable with the botnet, making it possible for botmasters to rent out a single RBnet for multiple criminal redirection infrastructures. Finally, as a side effect of proxy bots' poor utilization of resources, botmasters must be more diligent in management. They must ensure that enough bots are online and that they are intelligently dispersed across multiple DNS servers, such that the number of victims connecting to an individual bot is within its ability to function as a proxy.

For the above reasons, we propose a system called *Redirection Botnet Seeker*, or *RB-Seeker*, that detects RBnets based on their intrinsic network behavior patterns. Our contributions are three-folded. First, the system make uses of comprehensive and abundant data sources, including approximately two month's worth of: NetFlow records dumped from a core router of the campus network, spam emails from online and local spam archives, and DNS logs from two major local DNS servers. These rich, real-world data sources complement each other and provide the RB-seeker with comprehensive views from multiple vantage points, resulting in better a detection rate of RBnets. Second, we design and develop several effective algorithms to exploit each unique features of RBnets, such as their connection patterns, flow characteristics, DNS record behavior and typical involvement in the spam/phishing attacks. As a practical implementation for enterprise networks, the RB-Seeker comprises multiple network-monitoring subsystems, collaborating to identify malicious redirection infrastructures. The first subsystem takes advantage of the fact that redirection is often used for phishing/advertising. It analyzes embedded HTTP links in spam emails acquired from various sources using traditional spam detection systems and looks for links that redirect victims to different domains. The second subsystem improves the redirection detection capability by analyzing passive network traces at a large (i.e., of 40,000 students and several thousand faculty and staff) university core router and exploiting the statistical difference between the connection patterns of redirection and normal browsing. Together, these two subsystems compile a database of redirection domains, which is used by the third subsystem that actively polls and monitors the DNS query results for suspicious behavior. Third, we developed a 2-tier detection system that can detect both typical/aggressive and stealthy RBnets (i.e., those mimicking valid DNS behavior, such as CDNs), which are likely participating in multiple botnet activities. In addition, as a behavior-based approach, the RB-Seeker doesn't rely on the malware signatures and is thus more immune to traditional evasion techniques that are often and successfully employed by botmasters (e.g., polymorphism or malware packer).

The remainder of the paper is organized as follows. We review the related literature in Section 2. Section 3 presents an overview of system design and architecture. Sections 4–6 describe three closely interacting subsystems of the RB-Seeker that cooperatively achieve accurate identification of RBnets based on their unique network and DNS behavior.

Section 8 evaluates the effectiveness of the RB-Seeker and the paper concludes with Section 9.

2 Related Work

Botnets have now become one of the biggest threats to the Internet community. Most of the previous research focused on analyzing and understanding the operations and threats of botnets [23]. Cook *et al.* [14] studied the structure of botnets and highlighted the potential threats of peer-to-peer (P2P) botnets. They also showed that detecting botnets based solely on the C&C channel is not effective. Rajab *et al.* [28] constructed a distributed measurement infrastructure to measure the Internet Relay Chat (IRC) botnet activities and showed that botnets contribute the majority of unwanted traffic in the Internet. The botnet's diurnal properties are studied in [15] and used to model the propagation of botnets. More recently, P2P botnets appeared in the wild that use the P2P infrastructure as the C&C channel and, therefore, are more robust against node failures and difficult to be taken down. Grizzard *et al.* [18] analyzed the architecture and communication protocol of a most recent P2P botnet, Peacom (a.k.a. *storm worm*) [10]. A model for advanced hybrid P2P botnets has also been proposed in [31] which provides robust connectivity, control traffic dispersion, encryption, easy recovery and many other techniques that significantly improve the capability of P2P botnets in surviving the node failure and shutdown of C&C channels.

While there are numerous strategies to mitigate the effects of malware, most of them are ill-suited for combating botnets. Due to the modular nature of bots and the popularity of bot-development toolkits, it becomes fairly easy for even moderately skilled hackers to acquire botnets of their own, churning out numerous bot variants before up-to-date signatures can be generated for a signature-based anti-virus or intrusion detection system. In addition, sandboxes, honeypots, and honeynets [34], while useful for capturing and analyzing malicious binaries, incur much too long of a time delay (often involving human intervention) to be practical for botnet mitigation. To address these limitations, several botnet detection approaches have recently been proposed, trying to discover bonets based on the network or host behavior typical of most bots. Since the IRC protocol [24] is currently the most popular C&C protocol used by botnets, most approaches to date focus on using some aspects of the IRC protocol, such as traffic monitoring and identifying IRC C&C servers, for botnet identification. For example, Rishi [17] passively monitors IRC traffic for suspicious IRC nicknames, IRC servers, and uncommon server ports to detect bot-infected machines. Binkley and Singh [9] proposed detection of IRC-based botnets via TCP anomaly detection and IRC message statistics. More recently, BotHunter [19] uses IDS-driven dialog correlation based on IRC C&C communication and other common actions taken during the life cycle of a bot for detection. Meanwhile, Karasaridis *et al.* [26] proposed a wide-scale detection technique that looks for typical network-flow patterns between bots and their controllers to track and analyze botnets in a large tier-1 ISP. BotSniffer [20] identifies HTTP- and IRC-based C&C channels by capturing the coordinated and synchronized communication patterns in the C&C traffic. Unfortunately, because of the reliance on IRC- or HTTP-based C&C protocols for identifying botnets, these detection schemes can potentially be subverted using encrypted channels or customized C&C protocols (e.g., P2P, FTP, etc.).

In this paper, we propose a novel detection technique for discovering the botnets involved in redirection infrastructures. Our approach differs from previous work in that, instead of identifying the C&C channels which can be evaded by botnets utilizing modified or customized C&C protocols, we focus on the intrinsic behavior of RBnets. The behavior can be collected in real time from a variety of network-level traces, such as NetFlow, spam emails and DNS logs, regardless of the C&C protocols used by botnets. Similar network-level traffic analysis has also been widely used in both research projects and commercial products to combat malicious attacks such as DDoS and spams. For instance, the Internet Motion Sensor (IMS) [8] is a global monitor system that utilizes many distributed sensors to monitor traffic to the dark spaces and capture scanning traffic of worm propagation. The major purpose of IMS is to detect random-scanning worms and prevent scanning traffic from reaching the victims. Hence, it is not efficient for detecting botnets, because not all scanning traffic originates from botnets and botnets often propagate through other channels, such as email attachments (e.g., Peacomm bot), social engineering, browser vulnerabilities, etc. Cisco's Ironport [12] and P-cube [13] products investigate network (in particular SMTP) traffic as it passes through the systems. They use blacklist- or content-based filtering to detect spam emails and stop them before reaching the mail server. Because of the growing involvement of RBnets in spam/phishing campaigns [7], our approach can be easily integrated into these systems for practical deployment. The success of Ironport and p-cube in detecting incoming spam emails at the connection level will enable our approach to proactively detect RBnets and protect unsuspecting customers.

Different redirection techniques have mostly been used on redirection web spams, where attackers try to boost their rank in the search engine by presenting false content to a crawler for indexing and automatically redirect the users' browser to a different web page. Wu and Davision [33] studied the distribution of different redirection techniques. Chellapilla and Maykov [11] researched the prevalence of JavaScript redirection on the web and gave a detailed taxonomy of different JavaScript-based redirection techniques. They concluded that this type of redirection is most notori-

ous and hard to detect due to the versatility of JavaScript, which allows a number of obfuscation and dynamic-script-injection techniques. Wang *et al.* [32] built a system called "Strider Honeymonkey" to visit each page with a web browser and analyze the redirection behavior of malicious web servers. Their results showed that most malicious web sites use front-end servers to automatically redirect browser traffic to a back-end exploit server, specializing in exploiting client computers. Along the same line, Spamscatter [7] mines the URL links in the spam email and follows any redirection to reach the destination scam websites. They found that over 68% of scams adopt certain redirection techniques to protect the true destination servers. Because of the large-scale disposable nature of botnets, bots become ideal platforms for hosting redirection services. This is exemplified by the recent emergence of *fast-flux service network* (FFSNs) [27], which achieve high availability by rapidly changing the IP addresses associated with the fast-flux domains. In FFSNs, most of the nodes are bots whose purpose is to redirect the unsuspecting users to the destination web site hosting the phishing/scam contents or exploit codes. Holz *et al.* [22] studied the characteristics of FFSNs and developed detection algorithms that first extract URL links in the spam emails and then identify FFSNs based on the number of unique IP addresses in DNS queries and the number of unique AS numbers of those IP addresses. This is very efficient in capturing FFSNs whose behavior is drastically different from the normal cases. RB-Seeker differs from these in several ways. First, RB-Seeker detects RBnets by utilizing more comprehensive data sources, including NetFlow, DNS query logs and URLs in spam emails. This approach effectively alleviates the shortcomings of the spam-only approach. For instance, the URLs embedded a spam email could be heavily obfuscated or included inside a PDF or image. Furthermore, inspecting the content of all the emails is not always possible given privacy concerns. Second, RB-Seeker monitors multiple features of suspicious domains' DNS behavior over an extended period of time, utilizing an effective 2-tier detection strategy; this enables RB-Seeker to accurately detect RBnets with both aggressive and slow-changing (stealthy) DNS characteristics.

3 System Architecture

We have developed and prototyped a system aimed at the automatic identification of suspicious redirection behavior and bot-infected computers involved in RBnets. Fig. 1 shows the architecture of the proof-of-concept RB-Seeker, which consists of three cooperative subsystems, monitoring three primary input sources: spam emails (acquired from various sources using existing spam detectors), NetFlow data, and DNS server logs.

The first subsystem, called the *Spam Source Subsystem* (SSS), consists of two components: *content analysis* component and *URL probing engine*. The former accrues a vast collection of spam emails using traditional spam detection techniques: personal spam mailboxes, online spam archives, and a spam relay server setup on a residential network. It analyzes the spam and extracts the embedded links into the *spam URL database*. The URL probing engine follows the embedded links stored in the spam URL database and compiles a list of domains participating in redirection, which are added to the *redirection domain database*. The second component, the *NetFlow Analysis Subsystem* (NAS), also generates a list of redirection domains. However, unlike the SSS, the NAS monitors network flows on a large university core router and uses sequential hypothesis testing to detect IPs participating in redirection based on flow characteristics. These IPs are fed into the *correlation engine*, which uses DNS query logs to extract the associated domains, adding them to the redirection domain database. The redirection domain database compiled by the SSS and the NAS is used by our third—and final—subsystem, the *active DNS Anomaly Detection Subsystem* (a-DADS), which comprises two components. The first component, the *DNS probing engine*, continuously performs DNS digs on the domains in the redirection domain database, logging the results to the *DNS query database*. The other component of the a-DADS, the *RBnet classification engine*, extracts various attributes for each domain from the DNS query database and uses a hyperplane decision function to classify the domains as valid (i.e., benign) or malicious (i.e., belonging to a RBnet). If a domain is determined to be valid, it is removed from the redirection domain database and whitelisted to prevent the SSS or the NAS from reading it. When a RBnet domain is detected, it generates an alert report, containing the detailed DNS query logs for the domain (stored in the DNS query database), allowing for further manual analysis on the RBnet's DNS behavior if needed.

4 Spam Source Subsystem

Because spammers are driven by the incentive of profits, most spam emails contain embedded phishing or scam links and allure unsuspecting individuals to phishing/scam websites or web pages containing malicious exploit code. In most cases, to protect the destination servers, users are redirected through one-or-more redirection servers, which are likely to be compromised computers serving as RBs. Taking advantage of this close connection between RBs and spam/phishing emails, the SSS uses the embedded URLs in the spam message bodies as the starting point to trace out and detect machines participating in the redirection infrastructure. The SSS starts with the real-time collection of spam emails from multiple sources, including a spam relay

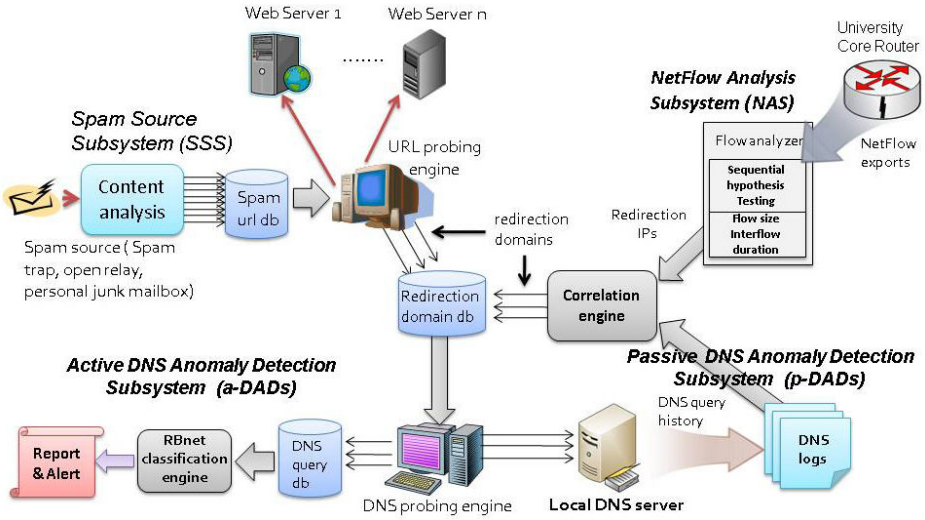

Figure 1: Architecture of the proof-of-concept RB-Seeker

trap[1] set up at a residential network, spam emails from the department mail server and several personal junk mailboxes at large email service providers. The SSS also downloads the latest spam emails from an online spam archive [21], which publishes more than 50,000 spam emails monthly. Upon collecting a new spam email, content analysis is performed on the message body to extract embedded URLs. After eliminating legitimate URLs using a precomposed whitelist, the SSS timestamps the remaining suspicious links and puts them into the spam URL database. The probing engine periodically retrieves URLs from the database and probes them to detect redirection behaviors.

According to [11], modern browsers can be redirected in three ways: HTTP-status-code redirection (e.g., 301 moved permanently, 302 temporary redirect, etc.), HTTP-meta-refresh-header redirection, and client-side script redirection. The SSS handles HTTP-status-code redirection by using wget [3] to fetch the webpage of a URL link. By default, wget detects the redirection status code and follows the URL specified in the location header entry of the HTTP response. By parsing the wget log file, the SSS can identify all intermediate redirection servers using HTTP-status-code redirection. Unfortunately, wget does not handle the other two types of redirection. To solve the problem, the SSS analyzes each downloaded webpage and searches for HTTP refresh tags and redirection scripts. More specifically, to capture HTTP-meta-refresh-header redirection (where the web pages use a META tag to redirect users), the SSS searches for the specific META tag with http-equiv attribute set to "refresh" and extracts the destination URL from the content attribute. For example, a typical use of META refresh header is as follows: `<meta http-equiv="refresh" content="0;url=http://www.destination.com">`. For script-based redirection, which is known to be most notorious and difficult to capture, the SSS focuses on detecting the most common JavaScript-based redirection techniques by scanning the web page for JavaScript code that changes a location property (e.g., "window.location," "location.replace," "document.location," "location.href," etc.) to the redirection destination URL. However, this type of static analysis can be evaded by sophisticated redirection techniques such as obfuscation, dynamic code injection and self-modifying JavaScript code [11]. We leave the dynamic analysis of web pages to identify redirection as our future work. However, a possible solution is to use a client-side honeypot (such as Capture-HPC [1]) that drives a real browser (with JavaScript enabled) to visit each suspicious URL and monitor the transition between web pages. After extracting the destination URL from either META head or JavaScript code, the SSS invokes the probing engine again on these links to identify further redirection attempts. This procedure is repeated until no further redirection is identified in the final destination web page or a predefined threshold is reached (to prevent an infinite loop). The numbers of observed web pages that use each redirection technique are: 61280 (54.1%) Status Code , 6639 (5.9%) Refresh Tag and 45285 (40.0%) JavaScript.

5 NetFlow Analysis Subsystem

This section describes the design of the second subsystem, the *NetFlow Analysis Subsystem* (NAS). Although

[1] A spam relay trap works like a spam honeypot. It appears to be an open SMTP server that allows anyone on the Internet to send email through it (we block the outgoing traffic, so no spam emails are sent). Because spammers extensively scan and exploit open relays to re-route their spam emails, our open relay trap is able to collect a large amount of spam every day (on average 7,900 spam emails/day).

quite useful, spam emails as a data source provide only a single vantage point with its own limitations. For example, spammers send image- or PDF-based spam emails to evade content-based filtering, so URL links might not appear as plain text in the spam body. Also, a user could be directed to a RB by clicking a link on a malicious web page, an IM message or many other ways. In addition, inspecting each email body is not always possible because of privacy concerns/laws. To complement the SSS and improve the detectability of RBnets, the NAS takes advantage of NetFlow records, identifying redirection servers without the need for packet content analysis.

5.1 Redirection behavior characterization

Currently, the NAS operates on the NetFlow records collected from a core router of a very large university (the University of Michigan) network and looks for suspicious redirection attempts of web traffic. *NetFlow* is a network protocol developed by Cisco for summarizing IP traffic information [5]. Although capturing and analyzing packet-level data can provide the highest accuracy, the associated cost is prohibitively high even for a medium-size network. As a result, NetFlow, as a light-weight alternative, has become the most widely-used technique for network monitoring, traffic accounting, etc. A *flow* is defined as a sequence of packets between a source and a destination within a single session or connection. A NetFlow record contains a variety of flow-level information such as IP protocol, source/destination IP and port, start and finish timestamps, and flow size. However, packet contents are not available, making it impossible to examine packet payloads and detect redirection behavior via HTTP status code or refresh headers. To address this limitation, we developed several redirection identification heuristics based only on the transport-layer information available in NetFlow data and the correlation of the traffic flow's size and timing behavior. The intuition behind these heuristics is that the behavior of visiting a redirection web server exhibits unique characteristics in terms of flow size, flow duration, and inter-flow duration, which are statistically different from normal, non-redirecting websites (see Table 1 for a detailed comparison) and can thus be used to capture redirection activities. In Table 1, to obtain the "ground truth" of redirection behavior, we collected a set of server IPs that have been determined by the SSS to perform redirection activities; we then use use tcpdump to capture all the packets between the SSS and those servers. In this way, we can build a database of redirection behaviors from the confirmed redirection server IPs and compute the values for each feature. Similarly, the values for normal browsing are computed using two days' packet traces of a user's normal web browsing activities after removing the packets of identified redirection connections. Next, we will elaborate on each feature and the intuition behind it.

		Mean	Median	Std dev
Flow duration (ms)	redirection	305.5	128.6	2159.2
	normal	33042.3	10028.8	91912.5
Inter-flow duration (ms)	redirection	392.7	154.4	872.4
	normal	40132.9	1345.5	87281.0
Flow size (bytes)	redirection	2401	629	44530
	normal	51495	4852	192431

Table 1: Comparison of average flow characteristics between redirection and normal browsing

Short inter-flow duration Redirection often leads to multiple, consecutive HTTP flows from the same source IP address to different destination web servers within a short time period. The *inter-flow duration* is defined as the difference between the start times of two consecutive flows originating from the same source IP and destined for distinct destination IPs. Intuitively, the fast and automatic transition caused by redirection from one web server to another is in stark contrast to the considerably longer time taken for a user to move between websites during normal web browsing, e.g., by manually clicking the links. From Table 1, we can see that normal browsing usually takes two orders-of-magnitude longer than the redirection.

Small flow size The flow size of visiting a redirection website is much smaller than that of visiting a normal website. This is because the redirection server usually returns only the redirection command data, such as HTTP status code, so that it will not waste bandwidth and hence, can be used for redirecting more clients. For example, in the case of most HTTP-status-code redirections, the redirection server returns only several tens of bytes, containing the status code (e.g., 301, 302) and a new destination server location. On the other hand, visiting a normal website usually necessitates downloading its homepage (often with pictures, longer texts, and embedded objects); thus, the flows of normal browsing are much larger in size (see Table 1).

Short flow duration Because of the small amount of data returned by a redirection server, the communication time (i.e., flow duration) between the user and the redirection server is often much smaller than that for a valid web server. Because the purpose of a RB is to forward a client to the mothership hosting the nefarious contents, it is of no benefit for a RB to maintain the connection with the client longer than needed. In most cases, the RB terminates the connection as soon as the client is handed over to another web server. By contrast, the connection time in a normal web browsing is considerably longer, especially because the current version of HTTP/1.1 introduces the keep-alive mechanism, which allows long-lived, persistent connections. For example, Internet Explorer (IE) times out a connection only after 60 seconds of inactivity.

5.2 Sequential hypothesis testing

Based on the above characteristics, the NAS exploits the temporal and size correlations among flows to identify redirection behavior. The NAS first sorts flow records chronologically and groups them by the source IP addresses. Within each group, the NAS computes the values of each feature—inter-flow duration, flow size and flow duration—for each destination IP; this forms an observed sample for each connection event between the source IP and a destination server. Our goal is then to classify whether the remote server is performing "redirection" or "normal behavior." The simplest way is to set up a fixed threshold for all three features and make decisions based on each individual observation. However, as we will show later, the distributions of normal and redirection behaviors for all the features are very heavy-tailed, indicating that a simple threshold method may introduce significant classification errors. Intuitively, this could be improved by utilizing multiple observations so that each decision is made with a high level of confidence. To achieve this goal, we adopt the Sequential Probability Ratio Testing (SPRT) [30], a type of statistical hypothesis testing where the number of observations required by the test is not pre-determined, but is a random variable determined by the underlying distribution. In other words, a decision is made only after enough evidence has been accumulated to support the acceptance or rejection of the hypothesis. SPRT thus achieves high accuracy and has been widely used in many anomaly detection scenarios such as detecting port scanners [25] and botnets [20].

To perform the Sequential Hypothesis Testing (SHT), we consider two hypotheses: H_0 (the remote server is a normal server) and H_1 (the remote server is a redirection server). In order to demonstrate how SHT works, let's examine how the NAS uses it to arrive at a classification decision for the inter-flow-duration feature (the procedure is identical for the other two features). Assuming the hypothesis H_i holds, the inter-flow duration follows some distribution (how to model such a distribution is discussed in the next subsection) whose density function is denoted as $f_i(T_{inter}) = f(T_{inter}|H_i)$. Let $T_1, T_2, \ldots, T_n$ be a sequence of observed samples of the inter-flow duration for the same destination IP. We can compute the likelihood ratio as $\Lambda(n) = \frac{f_1(T_2)f_1(T_2)\cdots f_1(T_n)}{f_0(T_2)f_0(T_2)\cdots f_0(T_n)} = \prod_{k=1}^{n} \frac{f_1(T_k)}{f_0(T_k)}$. Then, for each stage k, or the k-th observation ($k = 1, 2, \ldots, n$), the test leads to one of three decisions based on the following decision rules: (1) accept H_1 if the likelihood ratio exceeds the threshold η_1; (2) accept H_0 if the likelihood ratio is below another threshold η_0; and (3) otherwise, pend and wait for another observation. More specifically, for the k-th observation of a new connection,

$$\text{Output} = \begin{cases} \text{Accept } H_1 & \text{if } \Lambda(n) \geq \eta_1 \\ \text{Accept } H_0 & \text{if } \Lambda(n) \leq \eta_0 \\ \text{Pend} & \text{otherwise.} \end{cases}$$

One nice property of SHT is that the thresholds η_0 and η_1 can be set according to the target false-positive rate α (type-1 error: reject H_0 although it is true) and false-negative rate β (type-2 error: accept H_0 although it is false). Wald [30] showed that, by setting the threshold to $\eta_0 = \frac{1-\beta}{\alpha}$ and $\eta_1 = \frac{\beta}{1-\alpha}$, the true false-positive and false-negative rates will deviate from α and β by only a small margin.

5.3 Flow-based redirection identification

Fig. 2 shows the flowchart of how the NAS combines the three features and applies SHT to detect redirection servers. When a new connection is observed from a source IP (assuming the new connection is the n-th observation), the inter-flow duration T_n (defined as time difference between the current flow and the immediately preceding flow from the same IP) is compared against a loose threshold; this threshold value is chosen so that any inter-flow duration larger than this threshold is very unlikely to have been caused by redirection.[2] Then, if the inter-flow duration is below the threshold, the hypothesis testing history $\Lambda(n-1)$ is retrieved from the database, and the new likelihood ratio is computed as $\Lambda(n) = \Lambda(n-1) * \frac{f_1(T_n)}{f_0(T_n)}$. Depending on the likelihood ratio, the NAS outputs one of three decisions: accept H_0, reject H_0 (i.e., accept H_1), or pend. If the output is to accept H_0, then the destination IP of the preceding flow is considered a normal server, and the hypothesis testing history for that IP is cleaned up. If the existing data samples cannot provide enough confidence to reject or accept the hypothesis, the pending decision is chosen, and the current likelihood ratio is stored in the *hypothesis testing database* for future testing when additional observations become available. Finally, if the output suggests that a potential redirection behavior has been observed (i.e., to accept H_1) according to the inter-flow duration, a second hypothesis testing is performed on the flow size of the preceding flow. The reason why the NAS relies on multiple metrics to identify redirection behavior is that a single metric often leads to false positives. Specifically, the inter-flow-based hypothesis testing cannot distinguish concurrent flows from redirection flows. Concurrent flows occur when the destination web server references resources (e.g., pictures, videos) from other servers. As a result, when the client browses the web page, it requests several concurrent connections to multiple destinations within a short time frame. This results in short inter-flow durations that are indistinguishable from those caused by redirection behavior. Thus, we use flow size as a second-line filter to eliminate the potential false positives resulting from concurrent flows. Because the purpose of concurrent flows is often to fetch the (multimedia) contents of a web page, the flow size is expected to be much larger than is needed for redirection commands (e.g., status code). The hypothesis testing on flow size determines

[2] In our current experiment, we set this threshold to 30 seconds.

whether to accept the hypothesis or store the likelihood ratio for future testing. If the result indicates a redirection behavior, then a third, optional, hypothesis testing on flow duration could be performed. The flow duration is optional because our experimental measurements have shown that some redirection servers do not terminate connections—even after sending the redirection status code. The idle connection is kept alive without any data transmission until the client browser times out and closes the session. We conjecture this could be due to misconfiguration of the server. Thus, if a more strict detection algorithm is desirable, the optional flow-duration hypothesis testing can be performed to reduce false positives at the cost of increasing the false-negative rate (e.g., the NAS may fail to detect long-lived redirection servers). In our current implementation of the NAS, only the first two hypothesis tests are performed.

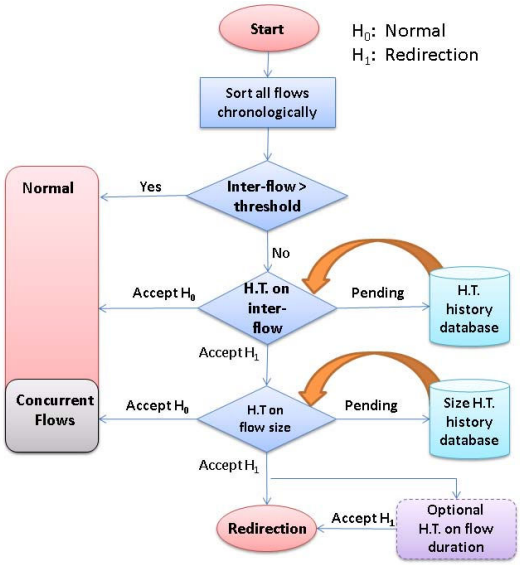

Figure 2: Flowchart of the algorithm for identification of redirection behaviors

5.4 Modeling the distribution of flow features

One of the pre-requisites for a hypothesis test is to determine the density function of different features conditioned on the hypothesis, i.e., $f_i(T)$ and $f_i(S)$, where T is the inter-flow duration, S is the flow size and $i = 0$ or 1. As mentioned before, to obtain the "ground truth" of redirection behaviors, we collect packet traces of confirmed redirection servers from the SSS and normal web-browsing activities to build two (i.e., normal and redirection) datasets for each feature. A simple examination of the histogram of these data sets shows that all the features follow non-negative heavy-tailed distributions, each with a single tail. Statistical distributions that satisfy these conditions include Pareto, log-normal and Weibull distributions. We apply the maximum likelihood (ML) method to estimate parameters for each distribution and compute Kolmogorov-Smirnov statistics [4], a popular method to evaluate how well a distribution fits the actual data. The result shows that the log-normal distribution achieves the best fit between the empirical data and analytical model. Its density function is given in the form of: $f(x;\mu,\sigma) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{(lnx-\mu)^2}{2\sigma^2}}$. The log-normal distribution is characterized by two parameters μ and σ. Table 2 shows the ML estimates of these two parameters and their 95% confidence interval for inter-flow and flow-size features. Fig. 3 depicts the CDF of inter-flow durations in both redirection and normal cases as well as the log-normal distribution fitting the results of the ML estimation. The flow-size result is similar to this, and hence omitted. Having estimated μ and σ, the hypothesis tests on these features can be done easily by calculating the likelihood ratio with the density function of a log-normal distribution and parameter values in Table 2.

	μ	95% C.I. of μ	σ	95% C.I. of σ
Inter-R	5.270	[5.260, 5.281]	0.974	[0.966, 0. 9812]
Inter-N	7.982	[7.896, 8.067]	2.512	[2.454, 2.574]
Size-R	6.529	[6.517, 6.542]	0.956	[0.948, 0.965]
Size-N	8.423	[8.380, 8.466]	2.093	[2.063, 2.125]

Table 2: Maximum likelihood estimates of parameters for a log-normal distribution (Inter-R means inter-flow duration for redirection, and Inter-N means inter-flow duration for normal browsing. Similarly, size-R(N) is defined.)

5.5 DNS log correlation

Using continuous monitoring of traffic flows, the NAS performs SHT to detect potential redirection activities and stores the IP addresses of suspected redirection servers (Fig. 1). However, many redirection servers could be benign, since redirection is also frequently used for legitimate purposes (e.g., web site migration, the use of a short and easily-remembered domain name to replace a long and convoluted one, redirection among alias domain names, etc.). To pinpoint malicious RBnets, we need to validate the DNS behavior of their domain names. However, NetFlow records only store the flow IP addresses without their DNS names. Note that the reverse DNS lookup is not useful in identifying the domain names for RBs; the forward mapping between the phishing/scam domains and bots' IPs are registered by the adversaries and are resolved by DNS servers they possibly control. Attackers can thus associate an arbitrary domain name with the bot's IP. On the other hand, a reverse DNS lookup returns the actual domain name of the RB as determined by the bot's ISP; thus, it will not match the malicious domain used in the scam. To address this problem, the NAS correlates the redirection IPs it has detected with domains found in the local DNS servers' DNS query logs. These identified redirection domains will first

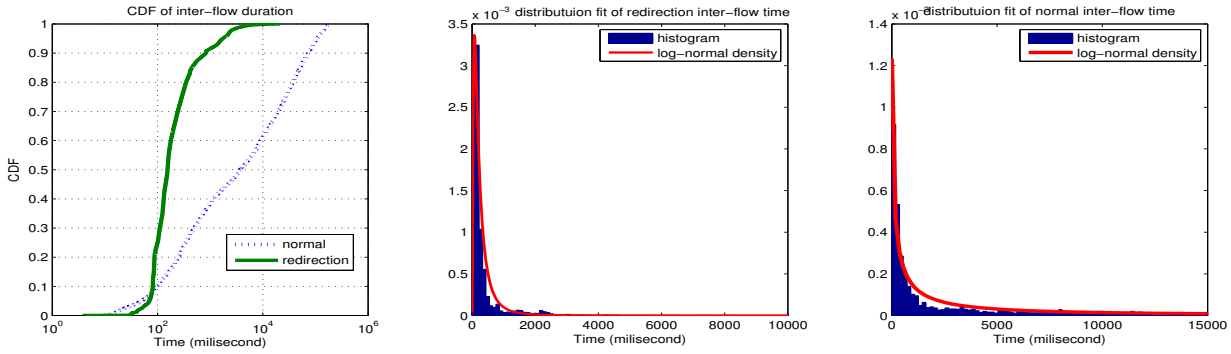

Figure 3: Log-normal distribution fit for inter-flow durations

be filtered against a known whitelist to remove legitimate redirection domains, such as popular content distribution networks[3] (e.g., Akamai, CoDeeN, LimeLight, etc.) and known redirection service domains [2] (e.g., google, yahoo, tinyurl, etc.). The remaining domains are placed into the redirection domain database to be probed and verified by the a-DADs, as we discuss next.

6 Active DNS Anomaly Detection Subsystem

The SSS and the NAS identify domains involved in redirection, either deterministically (from spam emails) or probabilistically (from NetFlow records), and store them in the redirection domain database. However, since valid domains commonly make use of redirection (e.g., to balance server load), there is no guarantee that the redirection domains detected belong to a RBnet. It is the purpose of the *active DNS Detection Subsystem* (a-DADS) to determine if any of the suspicious domains in the database actually belongs to a RBnet.

6.1 Data collection and analysis

For each unique domain in the redirection domain database, the a-DADS continuously performs and logs DNS queries for the domain's IPs (A records), name servers (NS records), name servers' IPs (NS-A records), the reverse DNS lookup on any IPs returned (i.e., the A and NS-A records), and the Autonomous System Number (ASN) to which each IP belongs. To analyze RBnet behavior over time, we continue to perform these digs until we have obtained at least a week's worth of *valid queries*: non-cached queries that didn't time out.

6.2 Characterization of RBnet behavior

RBnets, by their very nature, exhibit atypical DNS behavior. This is due to the way a RBnet is structured and the function it serves. A criminal, utilizing a redirection infrastructure for misdirection, will register an arbitrary domain name—perhaps a misspelling of a popular domain or an innocuous name used for a phishing email—and then point it to several bots in the RBnet. Thus, when the victim tries to visit the malicious domain, the DNS server will respond with one of the many bots' IPs, redirecting the victim. In order for this mechanism to provide reliable content delivery for the malicious domain, the botmaster must make certain the bots registered with DNS for the malicious domain are online. Otherwise, the victim will not be able to connect to the bot and be redirected to the nefarious content. Botnets naturally suffer from unreliable connectivity, since they are typically comprised of less secure home computers which are not always online. Even with increased use of 'always-online' broadband Internet, home desktops and laptops are often turned off or suspended, making them unreliable. To overcome this shortcoming, the botmaster must take certain measures to ensure the domain resolves to one of the online RBs, resulting in abnormal DNS query behavior. Based on the mechanisms available to the botmaster through DNS, we expect to observe behavioral abnormalities for the following attributes.

IP usage Botmasters incorporate several IP management strategies when advertising their RBnets to the DNS. These strategies cause the DNS query results for RBnet domains to exhibit discernable variations from those of valid domains using Content Distribution Networks (CDNs) or Round-Robin DNS (RRDNS). First, we expect there to be more unique IPs associated with a particular RBnet domain over time. RBnets will accrue, over time, more unique IPs than valid domains, since valid domains will have more stable servers hosting the content. In addition to supplying more IPs than valid domains over time, we expect many RBnets to supply more unique IPs per individual *valid query*. By supplying a larger set of IPs per query, the botmaster helps ensure the malicious domain resolves to a valid IP. With a larger pool of IPs, there is a higher probability one of them belongs to an online bot, decreasing the level of vigilance required in monitoring the RBnet's connectivity. As a further consequence of poor connectivity, botmasters will have

[3]We also developed an effective heuristic to detect previously unseen CDN domains and IPs, which will be discussed in Section 6.3.

to replace the IPs registered to their malicious domain frequently, requiring short TTL values. While CDNs also replace their IPs frequently, they will have a smaller pool of unique IPs over time than RBnets.

Reverse DNS lookup This involves a reverse DNS lookup on the IPs returned in the A records. While a reverse DNS lookup doesn't always return a result, when it does, it can be used to help detect a RBnet. Specifically, RBnets will often return names with "bad words" typical of home computers, such as cable, broadband, comcast, charter, dialup, dynamic, etc. Therefore, for each domain, we rank the occurrences of suspicious words. This is a reliable metric, since the reverse DNS name returned by a DNS server cannot easily be faked by a botmaster. For this reason, compromised home computers will often return reverse DNS names littered with suspicious words not present for valid domains (both RRDNS and CDNs). We also filter out valid domains containing "bad words" (e.g., comcast.net, charter.net), so that these are not unfairly weighted.

AS count Because the compromised computers that make up a botnet are scattered geographically, the IPs returned for RBnet domains will belong to a more diverse set of Autonomous Systems (ASes). Thus, we keep track of the number of unique ASes associated with a domain, as this should be a helpful metric in identifying RBnets.

6.3 CDN Filter

Consisting of thousands of servers distributed around the globe, CDNs must assume that any (and potentially many) of their servers could experience downtime due to network, software, or hardware failures. With this pragmatic view in mind, CDNs have been developed to be resilient to such failures, ensuring reliable content availability to their customers [6]. Consequently, they utilize DNS-based solutions similar to those currently being employed by botmasters. For example, CDNs and RBnets both use very small TTL values, allowing their networks to quickly respond to failure. Also, they both often advertise multiple IP addresses for a given domain, hedging their bets should some IP addresses go offline. CDNs also make use of aggressive load balancing, frequently changing the IPs advertised by DNS to ensure the highest throughput for their customers. These techniques often make the DNS behavior of CDNs appear akin to that of a RBnet.

Because of this behavioral similarity among CDNs and RBnets, we have developed a *CDN Filter* for the a-DADs to remove—from the redirection domain database—those domains that we can determine to be using legitimate CDNs. The CDN Filter operates based on the following two observations: (1) RBnet domains do not return IPs for legitimate CDNs in their DNS A records, and (2) each CDN server (with a corresponding IP) will be used to service multiple legitimate domains. As a result, the a-DADs CDN Filter analyzes the reverse DNS lookup of the A-record IPs for all the domains in the redirection domain database. When an A-record IP displays a reverse DNS name matching that of a legitimate CDN, we add the IPs seen for that domain to the *CDN-IP database*. The CDN-IP database is then cross-referenced against the IPs seen for other domains in the redirection domain database. When a domain is discovered to contain an IP from the CDN-IP database, it is flagged as using a CDN, and its IPs are added to the CDN-IP database. This process repeats, filtering out those domains that are using valid CDNs. In this way, the a-DADs CDN Filter removes those valid (non-RBnet) domains from the redirection domain database that exhibit DNS patterns most similar to those of RBnets.

6.4 RBnet classification

After filtering out the known CDNs with the CDN Filter, the a-DADS employs a *2-tier* linear Support Vector Machine (SVM) detection strategy on the remaining suspicious domains. The first-tier SVM (SVM-1) is designed to quickly identify those RBnets exhibiting a strong deviation from normal DNS behavior, which we term *Aggressive RBnets*. Any domain not identified as a RBnet by SVM-1 is further analyzed. The a-DADS continues to perform digs on the domain and applies the second-tier SVM (SVM-2) to the results. While SVM-1 is designed to identify Aggressive RBnets quickly from minimal valid queries, SVM-2 takes more time and is capable of detecting RBnets that try to mimic the short-term DNS behavior of valid domains, which we term *Stealth RBnets*.

Both SVM-1 and SVM-2 make use of a linear classifier of the form:

$$F(x) = \begin{cases} w^T x - b > 0 \text{ , if } x \text{ is valid domain} \\ w^T x - b < 0 \text{ , if } x \text{ is RB domain} \end{cases}$$

where w is a weight vector, b is a bias term, and x is a vector of behavioral attributes. These variables and vectors are different for each tier, and will be discussed next.

6.4.1 SVM-1

Using the CDN Filter, we filtered out any known CDN domains from the redirection domain database compiled by the SSS and the NAS. After filtering, the remaining suspicious domains were predominantly RRDNS, with a few CDN domains that escaped detection by our filter. We carefully selected a set of 124 valid domains that were representative of the different types of valid DNS behavior we observed. We also manually identified 18 Aggressive RBnet domains, which were easy to identify by hand due to their aggressive IP management tactics. These 142 domains (124 valid domains and 18 Aggressive RBnet domains)

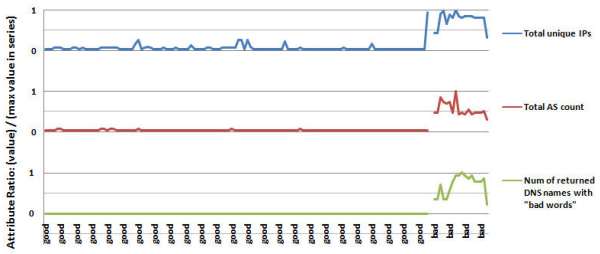

Figure 4: Domain attributes for the 142 domains in SVM-1 training set (two valid queries)

composed the *training set* for SVM-1. We then used 10-fold cross-validation on the training set to determine which behavioral characteristics best differentiated RBnets from valid domains based on only two valid queries. We discovered that three behavioral characteristics dominated the SVM equation: the total number of unique IPs seen, the total number of unique ASes seen, and the number of returned DNS names with "bad words". The other behaviorial characteristics we previously mentioned, while useful when analyzing multiple queries, were not as significant when observing only two valid queries. We chose to use the linear SVM best suited for classification based on the minimal number of valid queries, since the goal of SVM-1 is fast, accurate detection of Aggressive RBnets. The resulting equation is used for SVM-1, with the value returned being indicative of a domain's suspicion level:

$$\begin{aligned} f(x) &= w^T x - b \\ &= -1.257 * N_{unique\_IPs} - 26.401 * N_{ASes} \\ &\quad -13.024 * N_{DNS\_bad\_words} + 162.851 \end{aligned}$$

where N_{unique_IPs} is the number of unique IPs, N_{ASes} is the number of ASes, and $N_{DNS\_bad\_words}$ is the number of DNS "bad words" seen (should the reverse DNS lookup return a result). Testing the SVM-1 equation by using 10-fold cross-validation on the training set achieved 99.3% accuracy. We further evaluated the accuracy of SVM-1 by running it on the remaining domains in the redirection domain database *not* used in the training set; the evaluation of these results will be discussed Section 8.2.

A graph of the three attributes used in SVM-1 can be seen in Fig. 4. Each attribute is represented as a fraction of the largest value seen for that attribute among all the domains in the training set, allowing us to show all their relationships on a scale from 0 to 1. From Fig. 4, it is clear that Aggressive RBnets display a distinct behavioral difference from valid domains for the monitored attributes (gaps in the graph visually separate valid domains from Aggressive RBnet domains). The spike at the end of the good domains for the *Total unique IPs* is due to a CDN that managed to escape our CDN Filter. While it contains a large number of IPs (on par with Aggressive RBnets), there is a noticeable difference in its number of ASes and DNS "bad words". This difference allows SVM-1 to classify it as benign, causing it to be further monitored by SVM-2. Should any RBnets exhibit behavior similar to valid CDNs, they also will be monitored by SVM-2, which takes advantage of long-term DNS behavior to distinguish valid domains from RBnets.

6.4.2 SVM-2

One factor that will differentiate a Stealth RBnet from a valid domain or a CDN is the number of unique IPs and ASes it accrues *over time*. While DNS queries for valid domains (such as some CDNs) may return many IPs spanning multiple ASes (similar to RBnets), queries will continue to return those same IPs after a significant period of time. That is to say, the number of unique IPs and ASes returned by a valid domain over a day will be nearly the same set of IPs and ASes returned a few days later. This is because valid domains have fairly stable servers. While hardware or software failures may result in a server temporarily going offline, causing a new IP to be introduced to the DNS, they will not remain offline indefinitely. Ultimately, the problem will be fixed, the server brought back online, and its IP reintroduced into the DNS. On the contrary, Stealth RBnets are only able to *appear* like valid domains; they are still composed of compromised computers. The compromised computers may be more persistent than those in Aggressive RBnets, but they will still be more unreliable than the servers used in valid domains, such as CDNs. Additionally, some Stealth RBnets may utilize a legitimate redirection infrastructure that has been compromised, allowing them to mimic valid domains more easily. However, the servers in the compromised redirection infrastructure will still be less persistent than a valid CDN for the following reasons. First, the system administrators of the legitimate redirection infrastructure might thwart the botmasters' abuse of their system, rendering some of the botmasters' IPs useless. Second, in an effort to remain undetected by the system administrators, the botmaster will have to continuously change which servers are being exploited for the Stealth RBnet. In either case, the Stealth RBnet will slowly, over time, continue to accrue more and more unique IPs that span more and more ASes. This is in direct opposition to a valid domain or CDN, which has a fairly stable pool of server IPs to advertise to the DNS.

From our manual analysis of Stealth RBnets, we discovered that they tend not to return reverse DNS names. This could be because they are not composed of home computers (which tend to return reverse DNS names more often than legitimate servers), or are utilizing legitimate redirection infrastructures that have been compromised. Additionally, they tend to show very little variance in unique IPs and ASes between valid queries. We discovered this is because they are utilizing very short TTL values of around one second. This allows the botmaster to use a single IP (or a

small set of IPs) for multiple valid queries. The incredibly small TTL provides the botmaster with a fine level of control, permitting the IP to be changed as soon as the bot goes offline. In this way, the botmaster can keep both the unique IP count and the number of ASes low across multiple, valid queries, allowing the Stealth RBnet to go undetected by our SVM-1 as well as traditional FFSN detectors. To counter this strategy, our SVM-2 monitors the number of unique IPs and ASes seen in a day. It then continues to monitor the suspicious domain for up to a week, analyzing how many unique IPs and ASes it has accrued after this time span.

For the SVM-2's training set, we removed the 18 Aggressive RBnet domains from the SVM-1's training set and replaced them with 10 Stealth RBnet domains, which we identified manually. We then used 10-fold cross-validation on the 134-domain training set (124 valid domains and 10 Stealth RBnet domains) to determine the behavioral attributes best suited for differentiating Stealth RBnets from valid domains, given an extended observational period. As expected, the previously mentioned attributes based on changes between valid queries became insignificant due to the very short TTL value imposed by the botmaster. Additionally, the reverse DNS names with "bad words" became insignificant because none of the Stealth RBnets returned reverse DNS names. Thus, we found that SVM-2 only needed to monitor the number of unique IPs and ASes seen over time, for up to 1 week. We tested these metrics using 10-fold cross-validation on the training set and achieved 96.7% accuracy. We further evaluated the accuracy of SVM-2 by running it on the remaining domains in the redirection domain database *not* used in the training set; the evaluation of these results will be discussed in Section 8.2. The resulting linear equation is used for SVM-2, with the result indicating a domain's suspicion level:

$$\begin{aligned} f(x) &= w^T x - b \\ &= 52.497 * N_{DAY\_unique\_IPs} - 63.109 * N_{WEEK\_unique\_IPs} \\ &\quad -10.924 * (N_{DAY\_ASes} + N_{WEEK\_ASes}) + 227.985 \end{aligned}$$

where $N_{DAY\_unique\_IPs}$ is the number of unique IPs seen after a day, $N_{WEEK\_unique\_IPs}$ is the number of unique IPs seen after a week, N_{DAY_ASes} is the number of ASes seen after a day, N_{WEEK_ASes} is the number of ASes seen after a week. Fig. 5 shows a graph of these four attributes for a subset of SVM-2 training set. It is clear from the graph, that while some good domains slightly increase their total unique IP count from a day to a week, the increase is not nearly as drastic as with Stealth RBnets. Furthermore, all of the good domains have a constant number of ASes over the week, whereas most of the Stealth RBnets display a slight increase. Also, from Fig. 5, it is apparent that during the first day, the Stealth RBnets and the good domains share similar behaviorial attributes. It is only after monitoring for an extended period of time that the Stealth RBnets show their true colors, demonstrating the need for the more timely approach of SVM-2.

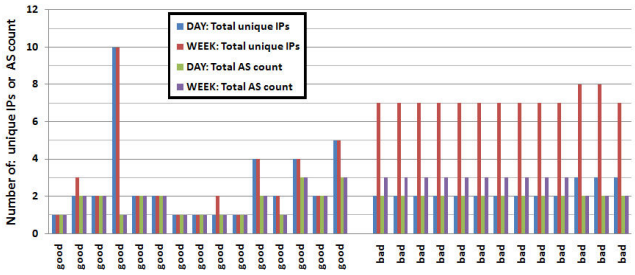

Figure 5: Domain attributes for subset of good and bad domains in SVM-2 training set

7 Discussion

Thus far, we have described the architecture of the RB-Seeker and its effectiveness in detecting current RBnets. However, security solutions are in a constant arms race between defenders and attackers, and the RB-Seeker is no exception. In this section, we discuss several ways adversaries may attempt to evade the RB-Seeker, providing potential countermeasures against them.

An attacker or botmaster who has learned the RB-Seeker's detection schemes may try to evade or confuse them by altering the RBnet's behavior according to the features used by the NAS, the SSS and the a-DADS. For instance, adversaries may try to confuse the NAS by invalidating the basic assumption that the NAS has made for redirection activities. Specifically, RBs may attempt to mimic the normal, non-redirection servers by waiting for an extended period of time (e.g,. 30 seconds) before redirecting clients, creating a longer inter-flow duration. They may also try sending useless content in their packets in addition to redirection commands, increasing the flow size. This may force the NAS to delay the detection decisions in order to accumulate enough observational samples. In the worst case, the NAS may mis-classify the redirection activities as normal. Like most behavior-based detection systems, the NAS is vulnerable to mimicry attacks: where adversaries successfully disguise their behavior as normal activities. However, because the characteristics of redirection are generally two orders-of-magnitude smaller than those of normal browsing (Table 1), in order to mimic the normal behavior, the attacker has to use most of the bot's already limited resources. For example, the bot must keep connections alive and send useless data, which will limit the number of victims that can be served by each individual bot. Otherwise, their consistent deviation from normal activities will still present a good chance of being caught by the NAS after observing enough samples. Second, to prevent the SSS from automatically extracting HTTP links from the email body, attackers may embed obfuscated/encoded URL links in spam emails instead of using plaintext or HTML format. They could also take advantage of sophisticated redirection techniques (e.g., obfuscated JavaScript) to circumvent the redi-

rection detection engine in the SSS. Although our prototype implementation only handles the most common and simple URL formats and redirection techniques, the SSS can be easily strengthened to counter such evasion tactics by incorporating existing methods for analyzing text embedded in images [16] and detecting sophisticated redirection links with client-side honeypots [1]. Finally, to circumvent the a-DADS detection, a botmaster may attempt to mirror the DNS behavior of popular CDNs by lowering the number and diversity of IPs associated with the domain. However, as discussed earlier, this not only limits the availability and throughput achievable by the RBnets, but these Stealth RBnets can still be detected with the a-DADS's improved CDN filtering technique and 2-tier detection strategy. Therefore, while there are several ways a botmaster could attempt to evade detection, some of them are too expensive to provide enough incentives for botmasters. Furthermore, as is demonstrated in the next section, the RB-seeker is still quite effective in identifying many RBnets.

8 Implementation and Evaluation

In this section we describe the implementation of the overall system and evaluate the overhead of its subcomponents. We then evaluate the performance of the a-DADS classification function, comparing it with the current state-of-the-art. Lastly, we briefly describe some of the DNS behavior for the RBnets detected with the RB-Seeker.

8.1 Implementation and overhead evaluation

We implemented a proof-of-concept RB-Seeker for Linux kernel 2.6.18 on an HP ServerBlade with 2 Dual-Core AMD Opteron(tm) Processors (2.2 GHz, 2024 KB cache), 4 GB of RAM, and 260 GB of disk space. The subcomponents were implemented in Perl and Python. They were continuously run to extract redirection domains from spam emails and NetFlow traces and perform DNS queries on the suspect domains.

On average, the SSS analyzes approximately 10,000 spam emails everyday (80% from spam relay and 20% from the spam archive and personal junk mail boxes) and extracts 9,000 unique URL links. Among them, the SSS applies the techniques described in Section 4 and determines more than 700 redirection domains everyday, adding them to the redirection domain database. Meanwhile, the NAS receives 95,000,000 flows from the core router everyday, 6,974,015 of which are HTTP flows[4] and are analyzed by the SHT algorithm (described in Section 5) to identify redirection activities. On average, the NAS identifies between 500 and 600 domains daily. We also test the processing speed of the NAS: the results show that the NAS is capable of parsing one day's HTTP flow data within 10 minutes, demonstrating its efficiency and suitability for online analysis. Another important factor that influences the NAS's speed in detecting a redirection server is the number of flows (i.e., observational samples) needed for the SHT algorithm to make a decision (i.e., accept or reject a hypothesis). Since the required number of observed samples in sequential testing is a random variable, depending on both current and historical observations [30], the expected number of observations (flows) for the NAS to determine if the destination IP is performing redirection can be approximated by:

$$E[N|H_1] = \frac{\beta \ln \frac{\beta}{1-\alpha} + (1-\beta)\ln \frac{1-\beta}{\alpha}}{E[\ln \frac{f_1(x)}{f_0(x)}]} = \frac{\beta \ln \frac{\beta}{1-\alpha} + (1-\beta)\ln \frac{1-\beta}{\alpha}}{\ln \frac{\sigma_0}{\sigma_1} + \frac{\sigma_1^2 + (\mu_1-\mu_0)^2}{2\sigma_0^2} - \frac{1}{2}}$$

where μ_i and σ_i $(i = 0, 1)$ are parameters for a log-normal distribution on the condition of normal browsing (H_0) and redirection (H_1). The values of μ_i and σ_i can be found in Table 2. As a result, the expected number of observations depends only on the target false-positive rate (α) or false-negative rate (β). Intuitively, if we want to reduce α and β, the expected number of required flows will increase, because the NAS has to accumulate more observed samples to reach the desired confidence level before making a decision. Figs. 8 and 9 in the appendix depict $E[N|H_1]$ with different values of α and β based on inter-flow duration and flow size, i.e., the expected number of flows the NAS has to observe in order to accept the hypothesis that the destination server performs redirection. One can see from the figures that the NAS is able to detect redirection servers by using only a small number of observed samples (normally 5 or 6) with low false-positive and false-negative rates.

In addition, because the a-DADS can be digging quite a few redirection domains simultaneously, we split its functionality into two parts to keep its overhead and memory footprint small. The first part simply reads the most recent domains in the redirection domain database (built by the SSS and the NAS) and performs digs on the domains, logging the results to the DNS query database. The second part then runs the RBnet classifier on these DNS logs once two valid queries have been obtained. If, based on these two queries, the classifier cannot identify the domain as belonging to a RBnet, it continues to gather DNS queries on the domains until it has accumulated enough for SVM-2 to reattempt classification. This approach reduces the amount of memory required by the a-DADS and maintains DNS query logs (in the DNS query database) for the suspicious domains should manual analysis be required later.

When calculating the suspicion level for a domain, the classifier must read all the domain's data from the DNS query database, extract the relevant behavioral characteristics from the data, and then use those characteristics in the SVM equation (either SVM-1 or SVM-2). To determine the overhead of the unoptimized, proof-of-concept RBnet clas-

[4] We consider a flow an HTTP flow if its destination port is 80 or 8080. Since no packet payload information is available, we are not able to detect HTTP flows using non-standard ports.

sifier, we did run-time performance tests for 150 domains,[5] consisting of 50 randomly-chosen domains from each of the following sets: Aggressive RBnet domains, Stealth RBnet domains, and valid (i.e., benign) domains. We measured the total time it took for the RBnet classifier to classify each domain; this includes such unoptimized operations as reading the data, parsing the data, extracting the characteristics, etc. Therefore, we also measured the amount of time it took for SVM-1 and SVM-2 to calculate the suspicion levels used for classification (after the relevant characteristics have been extracted from the DNS query data). The implementation of SVM-1 and SVM-2 leaves little room for optimization, unlike the rest of the RBnet classifier. We found that the RBnet classifier had an average run-time of 0.644 second per domain. Meanwhile, SVM-1 and SVM-2 had average run-times of 8 and 12 microseconds, respectively, hence making them suitable for a real-time detection system. Even when unoptimized, the proof-of-concept RBnet classifier takes (on average) less than a second per domain—sufficient for fast detection.

8.2 Evaluation of RBnet classifier

The a-DADS's RBnet classifier (described in Section 6.4) continuously monitored the 91,600+ suspicious domains in the redirection domain database detected by the SSS and the NAS over a period of approximately two months. Utilizing the CDN Filter, the a-DADS was able to determine 4,164 CDN domains (4,506 IPs) based on the reverse DNS name. Using the recursive iteration technique described in Section 6.3, this number was increased to 5,005 CDN domains (5,185 IPs), for a 16.8% increase in CDN domains (13.1% increase in IPs). The remaining domains were further filtered for known valid domains, using a technique similar to the CDN Filter. After filtering these domains, the a-DADS continued to monitor the remaining 35,500+ domains, applying SVM-1 and SVM-2.

With just two valid queries, SVM-1 was able to detect 125 Aggressive RBnet domains comprising 3,541 bot IPs. These Aggressive RBnet domains were then removed from the redirection domain database. The a-DADS continued to monitor the remaining domains in the redirection domain database, pruning the list as new CDN Filter data became available. After a week of monitoring, SVM-2 was able to identify an additional 156 Stealth RBnet domains, which comprised 249 IPs. Thus, the RB-Seeker managed to isolate a total of 3,790 bot IPs utilized by 281 unique domains for nefarious purposes. Further analysis revealed that 64 of the 125 Aggressive RBnet domains (51.2%) were used as proxies while the remaining 61 (48.8%) were used in a redirection infrastructure. All of the observed Stealth RBnets were involved in redirection infrastructures. SVM-1 experienced one false-positive (FP-rate $< 0.008\%$ for SVM-1), which has been removed from the previously-reported results. The false positive arose from SVM-1 misclassifying a valid *mozilla.org* domain used in a CDN for distributing releases. This single false positive results in the a-DADS's RBnet classifier (SVM-1 and SVM-2) having the low FP-rate of $< 0.004\%$. This false positive can be avoided with the addition of *mozilla.org* to the CDN Filter.

[5] All the domains in the test already had enough DNS query data in the DNS query database for the classifier to arrive at a decision. Therefore, these performance measurements don't reflect the time needed to gather the necessary DNS data, only the time taken to process it and calculate suspicion levels.

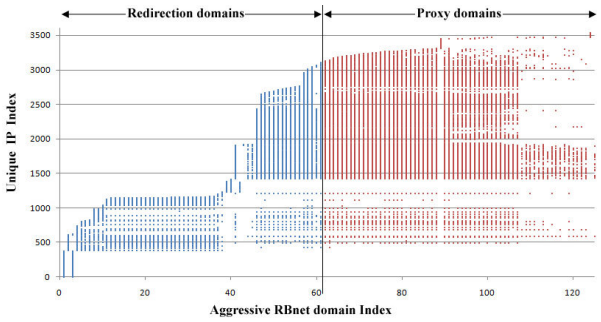

Figure 6: Unique IPs (as represented by unique IP index) seen for each detected Aggressive RBnet domain

For comparison, we also implemented the *FFSN detector* described in [22] and ran it on all the domains detected by the NAS and the SSS. The FFSN detector looks at the number of IPs and ASes over two valid queries. It succeeded in correctly identifying 124 of the 125 Aggressive RBnet domains that the RB-Seeker detected. It also incurred a false positive, incorrectly classifying the *mozilla.org* domain. Because the FFSN detector only looks at two valid queries, it is unable to distinguish the slower, more stealthy Stealth RBnets from valid domains (such as CDNs). The a-DADS's SVM-1 suffers from this same limitation. However, by utilizing the 2-tier system and monitoring suspicious domains over a longer period of time, SVM-2 is capable of detecting these Stealth RBnets.

8.3 Analysis of detected RBnets

Using the 125 Aggressive RBnet domains the RB-Seeker detected, we generated the plot in Fig. 6, which shows the unique 3,541 IPs (represented by a unique index) seen for each Aggressive RBnet domain. The graph is divided into two parts along the x-axis: the left half represents domains using redirection while the right half contains domains using proxies. Likewise, Fig. 7 shows a plot of the 156 Stealth RBnet domains and their corresponding 249 unique IPs (also represented by a unique index). The Aggressive RBnets and Stealth RBnets are shown in separate plots because they share no common IPs.

From the figures, it is apparent that while Stealth RBnets have more unique domains than Aggressive RBnets, they

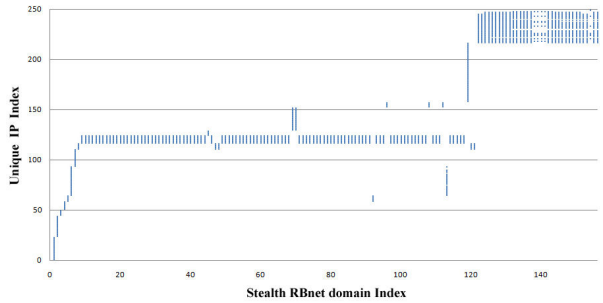

Figure 7: Unique IPs (as represented by unique IP index) seen for each detected Stealth RBnet domain

utilize vastly fewer IPs. Because Stealth RBnets attempt to mimic the DNS behavior of valid domains (such as CDNs), they must use computers with more stable IPs than Aggressive RBnets. If a Stealth RBnet attempted to use less persistent computers and still keep its DNS behavior "below the radar," the availability of the nefarious content serviced by the RBnet would suffer. It is the unreliable connectivity of the constituent bots in an Aggressive RBnet that necessitates the abundant advertising of IPs. Stealth RBnets, on the other hand, consist of much more reliable redirection servers—whether they are compromised machines or legitimate redirection infrastructures that are being exploited. However, this doesn't mean that Stealth RBnets are exactly like valid CDNs. Recall from Fig. 5, that over time, Stealth RBnets must continue to supply fresh IPs, either to avoid detection or as a result of being detected. Thus, to differentiate Stealth RBnets from valid CDNs, it is necessary to perform long-term DNS monitoring.

One thing Stealth and Aggressive RBnets do seem to have in common is their use of multiple domains resolving to a similar set of IPs. This can indicate (1) that a single botnet is servicing multiple scams, (2) that a single scam is utilizing multiple domains to evade detection, or (3) that those computers serving as bots are highly susceptible to attacks and have been incorporated into multiple botnets and their scams. It is likely that each of these scenarios plays a role in the observed behavior. However, from Fig. 7, it seems that the Stealth RBnets have IPs that are more compartmentalized to various domains' indices. This seems to favor the scenario that those domains sharing a set of IPs are aliases for the same scam, while those domains with different sets of IPs are partaking in different scams. For example, there are many domains that seem to share the same IPs with IP indices around 125. These domains are all derivations of *[XXX].bay.livefilestore.com*, where *[XXX]* is some random character string. *Norton Safe Web* identifies *livefilestore.com* as a high risk for Trojan, viruses, worms, spybots, identity theft, and phishing attacks [29], affirming that this Stealth RBnet is involved in numerous malicious activities and employing multiple domain aliases. Looking at Fig. 6, we can see that roughly half the Aggressive RBnet domains are using redirection (labeled *redirection domains* in the graph) while the other half are using proxies (labeled *proxy domains* in the graph). Interestingly, many of the same IPs are shared among both redirection and proxy domains. This lends credence to the scenario that the shared IPs belong to computers which have been compromised by more than one bot, so they are being used in multiple, separate botnet campaigns (i.e., proxy vs. redirection campaigns). Also, notice from Fig 6 that those Aggressive RBnet domains below domain index 40 share only a small set of IPs with the higher domain indices. They also seem to have a large subset of IPs that they share only among themselves. While we would expect a break like this to occur near the division between redirection and proxy domains, this occurs in the middle of the redirection domains. Those redirection domains with a domain index greater than 40 tend to share more IPs with the proxy domains. This seems to indicate that many of these computers are servicing multiple botnet campaigns. While this could be the result of multiple bots compromising the same machines or a botmaster using a single botnet for multiple scams, one thing is certain: by discovering these bots and blocking their IPs, we can potentially mitigate numerous botnet scams.

9 Conclusion

In this paper, we have designed and implemented a prototype system called the RB-Seeker for detecting the RBnets, which are increasingly used by attackers to redirect unsuspecting victim to web servers hosting nefarious content. The RB-Seeker achieves fast and automatic detection of RBnets, irrespective of their C&C protocol or structure, by employing several statistical correlation and classification techniques to analyze network traffic and DNS behavior. The system begins with two parallel subsystems, the SSS and the NAS, as first-line filters, cooperatively detecting redirection domains from multiple data sources. The NAS explores unique temporal/spatial features (e.g., interflow duration, flow size) of typical redirection activities so that redirection domains can be identified without expensive inspection of packet payload. The second-line detector, the a-DADS, exploits the atypical DNS query statistics of RBnets to distinguish between malicious and legitimate domains. Our evaluation of the RB-Seeker on real-world traces shows its capable of detecting both Aggressive and Stealthy RBnets with low false positives. Because of the prevalent and major role botnets play in redirection infrastructures, fast and automatic detection of such RBnets can not only protect users from phishing and scam websites, but also potentially deter many other malicious activities commonly perpetrated by botnets. Furthermore, the RB-Seeker is expected to be incrementally deployable and easily incorporated into existing security systems since its data sources are readily available in most enterprise networks.

References

[1] Capture-hpc. https://projects.honeynet.org/capture-hpc/.

[2] Free url redirection services. http://www.emailaddresses.com/email_url.htm.

[3] Gnu wget. http://www.gnu.org/software/wget/.

[4] Kolmogorov-smirnov test. http://www.physics.csbsju.edu/stats/KS-test.html.

[5] Netflow. http://www.cisco.com/en/US/products/ps6601/products_ios_protocol_group_home.html.

[6] M. Afergan, J. Wein, and A. LaMeyer. Experience with some principles for building an internet-scale reliable system. In *WORLDS'05: Proceedings of the 2nd conference on Real, Large Distributed Systems*, pages 1–6, Berkeley, CA, USA, 2005. USENIX Association.

[7] D. S. Anderson, C. Fleizach, S. Savage, and G. M. Voelker. Spamscatter: characterizing internet scam hosting infrastructure. In *SS'07: Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*, pages 1–14, Berkeley, CA, USA, 2007. USENIX Association.

[8] M. Bailey, E. Cooke, F. Jahanian, and J. Nazario. The internet motion sensor - a distributed blackhole monitoring system. In *NDSS*, 2005.

[9] J. R. Binkley and S. Singh. An algorithm for anomaly-based botnet detection. In *SRUTI'06: Proceedings of the 2nd conference on Steps to Reducing Unwanted Traffic on the Internet*, pages 7–7, Berkeley, CA, USA, 2006. USENIX Association.

[10] F. Boldewint. Peacomm.c - cracking the nutshell. http://www.reconstructer.org/, September 2007.

[11] K. Chellapilla and A. Maykov. A taxonomy of javascript redirection spam. In *AIRWeb '07: Proceedings of the 3rd international workshop on Adversarial information retrieval on the web*, pages 81–88, New York, NY, USA, 2007. ACM.

[12] Cisco System Inc. Ironport. http://www.ironport.com/ technology/ironport_antispam.html, 2007.

[13] Cisco System Inc. P-cube. http://www.p-cube.com/solutions/index.shtml, 2007.

[14] E. Cooke, F. Jahanian, and D. McPherson. The zombie roundup: understanding, detecting, and disrupting botnets. In *SRUTI'05: Proceedings of the Steps to Reducing Unwanted Traffic on the Internet on Steps to Reducing Unwanted Traffic on the Internet Workshop*, pages 6–6, Berkeley, CA, USA, 2005. USENIX Association.

[15] D. Dagon, C. Zou, and W. Lee. Modeling botnet propagation using time zones. In *Proceedings of the 13th Annual Network and Distributed System Security Symposium (NDSS'06)*, February 2006.

[16] G. Fumera, I. Pillai, and F. Roli. Spam filtering based on the analysis of text information embedded into images. *J. Mach. Learn. Res.*, 7:2699–2720, 2006.

[17] J. Goebel and T. Holz. Rishi: identify bot contaminated hosts by irc nickname evaluation. In *HotBots'07: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, pages 8–8, Berkeley, CA, USA, 2007. USENIX Association.

[18] J. B. Grizzard, V. Sharma, C. Nunnery, B. B. Kang, and D. Dagon. Peer-to-peer botnets: overview and case study. In *HotBots'07: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, pages 1–1, Berkeley, CA, USA, 2007. USENIX Association.

[19] G. Gu, P. Porras, V. Yegneswaran, M. Fong, and W. Lee. Bothunter: detecting malware infection through ids-driven dialog correlation. In *SS'07: Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*, pages 1–16, Berkeley, CA, USA, 2007. USENIX Association.

[20] 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.

[21] B. Guenter. Spam archive. http://untroubled.org/spam/.

[22] T. Holz, C. Gorecki, K. Rieck, and F. C. Freiling. Measuring and detectin fast-flux service networks. In *In Proc. network and Distributed System Security (NDSS) Symposium*, 2008.

[23] Honeynet Project. Know you enemy: Tracking botnets. http://www.honeynet.org/papers/bots, 2005.

[24] D. R. J. Oikarinen. Internet relay chat (irc) protocol. IETF, Request for Comments (RFC) 1459, May 1993.

[25] J. Jung, V. Paxson, A. W. Berger, and H. Balakrishnan. Fast Portscan Detection Using Sequential Hypothesis Testing. In *IEEE Symposium on Security and Privacy 2004*, Oakland, CA, May 2004.

[26] A. Karasaridis, B. Rexroad, and D. Hoeflin. Wide-scale botnet detection and characterization. In *HotBots'07: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, pages 7–7, Berkeley, CA, USA, 2007. USENIX Association.

[27] H. Project. Know you enemy: Fast-flux service networks. http://www.honeynet.org/papers/ff/fast-flux.html, 2007.

[28] M. A. Rajab, J. Zarfoss, F. Monrose, and A. Terzis. A multifaceted approach to understanding the botnet phenomenon. In *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 41–52, New York, NY, USA, 2006. ACM.

[29] Syamentec Corp. Norton safe web, report for livefilestore.com. https://safeweb.norton.com/report/show?name=livefilestore.com.

[30] A. Wald. Sequential tests of statistical hypotheses. *The Annals of Mathematical Statistics*, 16(2):117–186, June 1945.

[31] P. Wang, S. Sparks, and C. C. Zou. An advanced hybrid peer-to-peer botnet. In *HotBots'07: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, pages 2–2, Berkeley, CA, USA, 2007. USENIX Association.

[32] Y.-M. Wang, D. Beck, X. Jiang, R. Roussev, C. Verbowski, S. Chen, and S. King. Automated web patrol with strider honeymonkeys: Finding web sites that exploit browser vulnerabilities. In *In Proc. network and Distributed System Security (NDSS) Symposium*, 2006.

[33] B. Wu and B. Davison. Cloaking and redirection: A preliminary study, 2005.

[34] V. Yegneswaran, P. Barford, and V. Paxon. Using honeynets for internet situational awareness. In *HOTNETS*, 2005.

APPENDIX

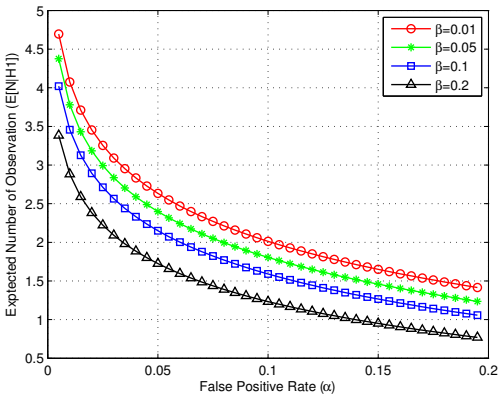

Figure 8: Expected number of flows required to determine redirection servers based on inter-flow duration

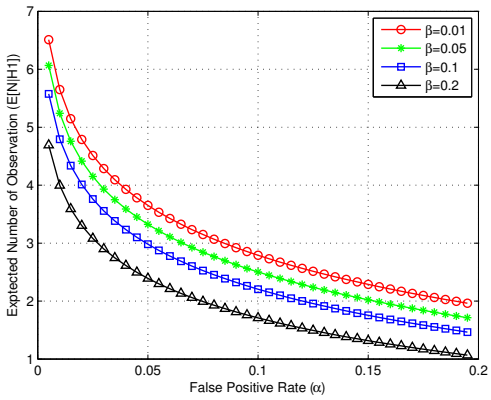

Figure 9: Expected number of flows required to determine redirection servers based on flow size