Measuring and Detecting Fast-Flux Service Networks

Thorsten Holz[1] Christian Gorecki[1] Konrad Rieck[2] Felix C. Freiling[1]

[1] University of Mannheim
Laboratory for Dependable Distributed Systems
Mannheim, Germany
{holz, gorecki, freiling}@informatik.uni-mannheim.de

[2] Fraunhofer FIRST
Intelligent Data Analysis
Berlin, Germany
konrad.rieck@first.fraunhofer.de

Abstract

We present the first empirical study of fast-flux service networks (FFSNs), a newly emerging and still not widely-known phenomenon in the Internet. FFSNs employ DNS to establish a proxy network on compromised machines through which illegal online services can be hosted with very high availability. Through our measurements we show that the threat which FFSNs pose is significant: FFSNs occur on a worldwide scale and already host a substantial percentage of online scams. Based on analysis of the principles of FFSNs, we develop a metric with which FFSNs can be effectively detected. Considering our detection technique we also discuss possible mitigation strategies.

1 Introduction

One of the most important properties of commercial service providers in the Internet is the continuous *availability* of their websites. If webservers are not online, the service cannot be offered, resulting in loss of profit. It is estimated that online shops like Amazon loose about $550,000 for every hour that their website is not online [20]. The main cause of unavailability have been *hardware faults*: Since electronic components have only a limited lifetime, computer and storage systems are prone to failures. Techniques from the area of *reliability engineering* [5], e.g., redundancy techniques like RAID [21] or commercial failover systems [12], help to overcome these failures. Nowadays, tolerance against the failure of individual hardware components is rather well-understood. However, *Distributed Denial-of-Service attacks* [17], especially in the form of network bandwidth exhaustion, pose a significant threat. This threat has become an almost daily nuisance with the advent of *botnets*, large networks of compromised machines which can be remote controlled by an attacker [9]. A medium-sized botnet of 1000 or 2000 machines is usually sufficient to take down almost any network service.

Several methods exist to alleviate the results of distributed denial-of-service attacks. We focus here on standard methods which use the global *Domain Name Service* (DNS). A well-known method is called *Round-robin DNS* (RRDNS) [7]. This method is used by large websites in order to distribute the load of incoming requests to several servers [8, 14] at a single physical location. A more advanced (and more expensive) technique is implemented by so called *Content Distribution Networks* (CDNs) like *Akamai* [1]. The basic idea is to distribute the load not only to multiple servers at a single location, but to also distribute these servers over the globe. The real benefit of CDNs comes with using DNS: When accessing the name of the service via DNS, the CDN computes with the help of sophisticated techniques the "nearest" server (in terms of network topology and current link characteristics) and returns its IP address. The client then establishes a connection to this server and retrieves the content from there. In effect, content is thereby moved "closer" to the client that sends the request, increasing responsiveness and availability.

RRDNS and CDNs are techniques employed by *legal* commercial organizations. Unfortunately, there are also a lot of *illegal* commercial organizations offering services in the Internet. Naturally, they also demand high availability for the services they offer. For example, a *spammer* that runs a website to sell pharmaceutical products, adult content, or replica watches can only make money if the website is reachable. As another example, consider a *phisher* that steals confidential information by redirecting users to fake online sites and tricking them into revealing their credentials. The phisher also requires the phishing website to be online most of the time; only then victims can fall for this scam. As a final example, consider a *botherder* who directs a large botnet. The botnet itself requires a reliable hosting infrastructure such that the botherder's commands can be sent to the bots or malicious binaries can be downloaded by existing bots or new victims.

The starting point of this paper is the question, how illegal organizations achieve high availability of their online

services, explicitly focusing on HTTP services (i.e., websites). The problems which such organizations face today is that law enforcement's abilities to take webservers with illegal content down has reached a certain level of effectiveness. RRDNS and CDNs are therefore no real alternatives for hosting scams. In a slightly ironic repetition of history, it seems that today illegal organizations are discovering classic redundancy techniques to increase the resilience of their infrastructure, as we explain in this paper.

In this paper we report on a newly emerging and still not widely-known threat in the Internet called a *fast-flux service network* (FFSN). Such a network shows a similar behavior as RRDNS and CDNs in regards to the network characteristics: A single service seems to be hosted by many different IP addresses. Roughly speaking, a FFSN uses rapid-changing DNS entries to build a hosting infrastructure with increased resilience. The key idea is to construct a distributed proxy network on top of compromised machines that redirects traffic through these proxies to a central site, which hosts the actual content. Taking down any of the proxies does not effect the availability of the central site: With the help of a technique similar to RRDNS, the attacker always returns a different set of IP addresses for a DNS query and thus distributes the traffic over the whole proxy network. This leads to an increased resilience since taking down such schemes usually needs cooperation with a domain name registrar. As we will see in this paper, a single fast-flux service network can consist of hundreds or even thousands of compromised machines.

Contributions. While the technology of FFSNs has been reported on elsewhere [28], it is still unclear how large the threat of FFSNs is and how they can be mitigated. In this paper we present the first empirical study of the fast-flux phenomenon giving many details about FFSNs we observed during a two-month period in summer 2007. By analyzing the general principles of these networks, we develop a metric with which FFSNs can be effectively detected using information offered by DNS. Such a metric can be used to develop methods to mitigate FFSNs, a topic which we also discuss in this paper. To summarize, the contributions of this paper are:

- We present the first detailed empirical study of FFSNs and measure with the help of several properties the extent of this threat. Our measurements show that almost 30% of all domains advertised in spam are hosted via FFSNs and we found several ten thousands of compromised machines during our study.

- We analyze the principles of FFSNs, from this develop a metric to detect FFSN and show the effectiveness of this metric by using measurements.

- We discuss mitigation strategies for FFSNs based on the developed metric.

Our contribution does not only clarify the background behind a fast growing threat within the Internet, but moreover we presents new results obtained after reinterpreting previous measurements [3] taking our findings into consideration. We are aware neither of any other work which has investigated FFSNs in an empirical way nor of work which discusses detection and mitigation strategies. Due to the severity of the novel FFSN threat, we feel that our results can also be helpful in practice.

Outline. In Sect. 2, we provide an overview of the technical background of RRDNS, CDNs, and FFSNs. We develop a metrical framework to automatically detect FFSNs based on their answers to DNS requests in Sect. 3 and show that our metric can identify FFSNs with high accuracy and almost no false positives. In Sect. 4, we provide detailed results of empirical measures for several FFSNs. Several strategies to mitigate FFSNs are proposed in Sect. 5 before we conclude the paper in Sect. 6 with a brief overview of future work in this area.

2 Technical Background

2.1 Round-Robin DNS

Round-robin DNS is implemented by responding to DNS requests not with a single DNS *A record* (i.e., hostname to IP address mapping), but a *list* of A records. The DNS server cycles through this list and returns them in a round-robin fashion.

Fig. 1 provides an example of round-robin DNS used by `myspace.com`, one of the Top 10 websites regarding traffic according to Alexa [2]. We performed the DNS lookup with `dig` [13], a tool dedicated to this task, and only show the `ANSWER` section for the sake of brevity. In total, three A records are returned for this particular query, all pointing to servers hosting the same content. The DNS client can then choose one of these A records and return the corresponding IP address. Basic DNS clients simply use the first record, but different strategies can exist, e.g., using the record which is closest to the DNS client in terms of network proximity. Every A record also has a *Time To Live* (TTL) for the mapping, specifying the amount of seconds the response remains valid. RFC 1912 recommends minimum TTL values of 1-5 days such that clients can benefit from the effects of DNS caching [4]. Shaikh et al. study the trade-off between responsiveness of round-robin based server selection, client-perceived latency, and overall scalability of the system and show that small TTL values can have negative effects [25]. If the DNS lookup is repeated

```
;; ANSWER SECTION:
myspace.com.            3600    IN      A       216.178.38.116
myspace.com.            3600    IN      A       216.178.38.121
myspace.com.            3600    IN      A       216.178.38.104
```

Figure 1: Example of round-robin DNS as used by `myspace.com`

while the answer is still valid, the query for `myspace.com` returns the same set of IP addresses, but in a different order. Even after the TTL has expired, i.e., after 3600 seconds in this example, a subsequent DNS lookup returns the same set of A records.

We tested all domains from the Alexa Top 500 list and found that almost 33 percent use some form of RRDNS, i.e., more than one A record was returned in a DNS lookup. Furthermore, we measured the TTL values used by these sites: about 43 percent of these domains have a TTL ≤ 1800.

2.2 Content Distribution Networks

Like RRDNS, CDNs also usually implement their service using DNS [10, 15, 24]: The domain name of the entity which wants to host its content via a CDN points to the nameservers of the CDN. With the help of sophisticated techniques, the CDN computes the (in terms of network topology and current link characteristics) nearest *edge server* and returns the IP address of this server to which the client then connects.

Fig. 2 depicts the A and CNAME (*canonical name*, an alias for one name to another) records returned in DNS lookups for `images.pcworld.com`, which uses Akamai to host its content. Again, the DNS lookup returns multiple A records which all belong to the network of Akamai. Compared to the previous example, the TTL is significantly lower. A low TTL is used by CDNs to quickly enable them to react to changes in link characteristics. The Akamai edge server is automatically picked depending on the type of content and the user's network location, i.e., it can change over time for a given end-user.

2.3 Fast-Flux Service Networks

From an attacker's perspective, the ideas behind round-robin DNS and content distribution networks have some interesting properties. For example, a spammer is interested in having a high reliability for hosting the domain advertised in his spamming e-mails. If he could advertise several IP addresses for a given domain, it would become harder to shut down the online pharmacy shop belonging to the scam: If at least one of the IP addresses returned in an A record is reachable, the whole scam is working. Moreover, a botherder is interested in scalability and he could use round-robin DNS to split the bots across multiple Command & Control servers in order to complicate shutdown attempts. In both examples, the resulting scam infrastructure would be more resilient to mitigation attempts.

RRDNS and CDNs return several IP addresses in response to a DNS request. As long as one of these addresses responds, the entire service is online. *Fast-flux service networks* (FFSNs) employ the same idea in an innovative way. The main characteristic of fast-flux is the rapid (*fast*) change in DNS answers: A given fast-flux domain returns a few A records from a larger pool of compromised machines ("flux-agents") and returns a *different* subset after the (low) TTL has expired. By using the compromised machines as proxies to route an incoming request to another system (control node / "mothership"), an attacker can build a resilient, robust, one-hop overlay network.

We explain the structure behind FFSNs with the help of a short example. The domain `thearmynext.info` was found in a spam e-mail in July 2007. The `dig` response for this domain is shown in the upper part of Fig. 3. We repeated the DNS lookup after the TTL timeout given in the first answer to have two consecutive lookups of the same domain. The results of the second lookup is shown in the lower part of Fig. 3. The DNS request returns five A records, similar to the round-robin DNS example above. However, all IP addresses belong to different network ranges. We performed a reverse DNS lookup, resolved the Autonomous System Number (ASN), and determined the country via geolocation lookup for each of the IP addresses returned in the first lookup. The results are shown in Tab. 1. Overall, we identify several interesting features: First, all IP addresses are located in DSL/dial-up network ranges located in several different countries, e.g., United States, Germany and Portugal. Second, the IP addresses belong to several different Autonomous Systems (AS). Third, the TTL is rather low with 600 seconds. And fourth, the DNS server returns a set of five totally different IP addresses in the second request.

A closer examination reveals that the A records returned by the DNS lookup point to IP addresses of suspected compromised machines which run a so called *flux-agent*. The flux-agents are basically proxies which redirect an incoming request to the *control node* [28], on which the actual content of the scam is hosted.

Fig. 4 illustrates the process of retrieving the content from a legitimate site: The client contacts the webserver and the content is sent directly from the server to the client.

```
;; ANSWER SECTION:
images.pcworld.com.                    900     IN     CNAME   images.pcworld.com.edgesuite.net.
images.pcworld.com.edgesuite.net.   21600     IN     CNAME   a1694.g.akamai.net.
a1694.g.akamai.net.                    20     IN     A       212.201.100.135
a1694.g.akamai.net.                    20     IN     A       212.201.100.141
```

Figure 2: Example of DNS lookup for domain `images.pcworld.com` hosted via Content Distribution Network, in this case Akamai

```
;; ANSWER SECTION:
thearmynext.info.       600     IN      A       69.183.26.53
thearmynext.info.       600     IN      A       76.205.234.131
thearmynext.info.       600     IN      A       85.177.96.105
thearmynext.info.       600     IN      A       217.129.178.138
thearmynext.info.       600     IN      A       24.98.252.230
```

```
;; ANSWER SECTION:
thearmynext.info.       600     IN      A       213.47.148.82
thearmynext.info.       600     IN      A       213.91.251.16
thearmynext.info.       600     IN      A       69.183.207.99
thearmynext.info.       600     IN      A       91.148.168.92
thearmynext.info.       600     IN      A       195.38.60.79
```

Figure 3: Example of A records returned for two consecutive DNS lookups of domain found in spam e-mail. The DNS lookups were performed 600 seconds apart

This is a common setup used by many websites.

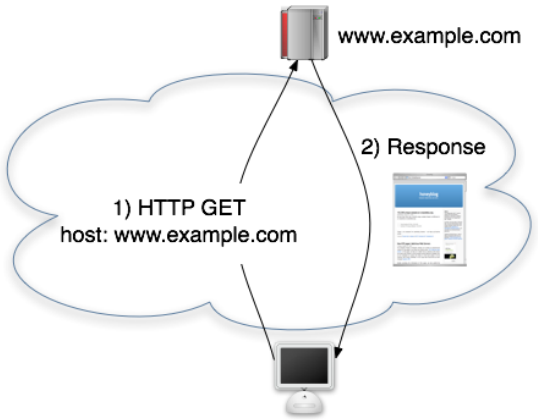

Figure 4: Content retrieval process for benign HTTP server

In a scam that uses FFSN for hosting, the process is slightly different (Fig. 5): The client uses DNS to resolve the domain and then contacts one of the flux-agents. The agent relays the request to the control node, which sends the content to the flux-agent. In the fourth step, the content is delivered to the client. Note that if the TTL for the fast-flux domain expires and the client performs another DNS lookup, the DNS lookup process will most likely return a different set of A records. This means that the client will then contact another flux-agent, but the request is relayed from that machine to the control node in order to retrieve the actual content. More technical details on fast-flux service networks can be found in a recent paper by the Honeynet Project [28].

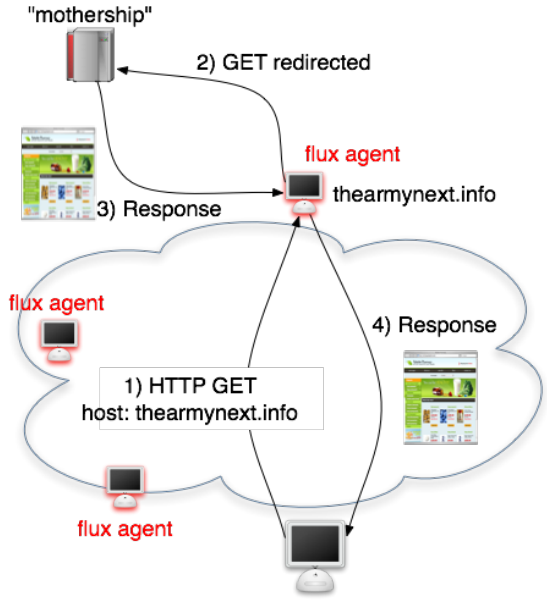

Figure 5: Content retrieval process for content being hosted in fast-flux service network

IP address returned in A record	Reverse DNS lookup for IP address	ASN	Country
69.183.26.53	69.183.26.53.adsl.snet.net.	7132	US
76.205.234.131	adsl-76-205-234-131.dsl.hstntx.sbcglobal.net.	7132	US
85.177.96.105	e177096105.adsl.alicedsl.de.	13184	DE
217.129.178.138	ac-217-129-178-138.netvisao.pt.	13156	PT
24.98.252.230	c-24-98-252-230.hsd1.ga.comcast.net.	7725	US

Table 1: Reverse DNS lookup, Autonomous System Number (ASN), and country for first set of A records returned for fast-flux domain from Figure 3.

3 Automated Identification of Fast-Flux Domains

As we want to distinguish between FFSNs and other legitimate domains in an automated way, we now turn to the extraction of features enabling us to decide whether a given domain is using the FFSN infrastructure or not.

Restrictions in establishing an FFSN. In contrast to legitimate service providers which may buy availability over CDNs, providers running FFSNs naturally suffer from two main restrictions:

- (IP address diversity) A scammer is not as free to choose the hardware and network location (IP address) of an individual node as freely as in a CDN. Basically, the FFSN has to live with those machines which can be compromised to run a flux-agent. The range of IP addresses must therefore be necessarily rather diverse and the attacker can not choose to have a node with a particular IP address.
- (No physical agent control) In contrast to CDNs which run in large computing centers which professionally host the servers and manage server failures through planned downtimes, a scammer does not have direct control over the machines which run the FFSN. Even worse, flux-agents usually run on ill-administered machines in dial-up networks which may go down any minute even if their uptime is rather large. This implies that there is no guaranteed uptime of the flux-agent the scammer can rely on.

Possible distinguishing parameters. Based on these two restrictions in establishing a FFSN, we now enumerate a set of parameters which can be used to distinguish DNS network behavior of CDNs from FFSNs. The absence of physical control over the flux-agents results in the consideration of the following two values:

- n_A, the number of unique A records returned in all DNS lookups: Legitimate domains commonly return only one to three A records, whereas fast-flux domains often return five or more A records in a single lookup in order to have a higher guarantee that at least one of the IPs is online.
- n_{NS}, the number of nameserver (NS) records in one single lookup: FFSNs can also host the nameserver within the fast-flux network [28] and often return several NS records and A records for the NS records. In contrast, legitimate domains commonly return a small set of NS records.

The restriction of IP address diversity results in the consideration of the following value:

- n_{ASN}, the number of unique ASNs for all A records: Legitimate domains and even the domains hosted via CDNs tend to return only A records from one particular AS. In contrast, FFSNs tend to be located in different ASs since the infected machines are scattered across different ISPs.

All the above parameters can be determined via DNS lookups and short post-processing of the result. Note that we do not consider the TTL value of the DNS entries as a good parameter. This is because legitimate domains like those hosted via CDNs have similar requirements as FFSNs with respect to the speed of adaptation to network congestion or server outages. The TTL value is, however, a good indicator to distinguish FFSN/CDN from RRDNS. Therefore we take only domains with a TTL of the A records below 1800 seconds into account, since higher TTL values can not be considered *fast* enough for rapid changes.

Fluxiness. In general, a metric to distinguish FFSNs from CDNs is a function of n_A, n_{AS}, and n_{NS}. Several possibilities to define this function exist. For example a first approximation could be the following value, which we call the *fluxiness* of a domain:

$$\varphi = n_A / n_{single}$$

The value n_{single} is the number of A records a single lookup returns. A value $\varphi = 1.0$ means that the set of A records

remains constant over several consecutive lookups, which is common for benign domains. In contrast, $\varphi > 1.0$ indicates that at least one new A record was observed in consecutive requests, a strong indication of CDNs and FFSNs. In the example of Fig. 3, $\varphi = 2.0$ since the second set of returned A records has no overlap with the first lookup.

Note that the fluxiness of a domain is implicitly contained in n_A and n_{ASN}: For FFSNs (and also CDNs), the number of observed A records (and thus potentially also number of ASNs) grows over time since the lookup process returns a different set of IPs over time.

Flux-Score. A general metric for detection of fast-flux domains can be derived by considering the observed parameters as vectors x of the form (n_A, n_{ASN}, n_{NS}). The resulting vector space enables definition of a linear decision function F using a weight vector w and a bias term b by

$$F(x) = \begin{cases} w^T x - b > 0 & \text{if } x \text{ is a fast-flux domain} \\ w^T x - b \leq 0 & \text{if } x \text{ is a benign domain} \end{cases}$$

The decision surface underlying F is the hyperplane $w^T x + b = 0$ separating instances of fast-flux service networks from benign domains.

Given a corpus of labeled fast-flux and benign domains, there exist numerous assignments of w and b correctly discriminating both classes, but differing in their ability to *generalize* beyond the seen data. A well-known technique for obtaining strong generalization is determining the *optimal hyperplane*, which separates classes with maximum margin [29]. For the linear case of the decision function F, an optimal hyperplane can be efficiently computed using the technique of linear programming [6].

Based on a labeled corpus of domains, we can determine a decision function F with high generalization ability by computing the weight vector w and bias b of the optimal hyperplane. The decision function F induces a scoring metric f for the detection of fast-flux domains referred to as *flux-score* and given by

$$f(x) = w^T x = w_1 \cdot n_A + w_2 \cdot n_{ASN} + w_3 \cdot n_{NS} \quad (1)$$

A flux-score $f(x) > b$ indicates an instance of a fast-flux service network, while lower scores correspond to benign domains. Furthermore, the flux-score provides a *ranking* of domains, such that higher values reflect a larger degree of fast-flux characteristics – implicitly corresponding to a larger distance from the optimal hyperplane of F.

Validation of current FFSN. To instantiate the weights w, we used empirical measurements of 128 manually verified fast-flux domains and 5,803 benign domains as input. The latter were randomly taken from the Open Directory Project [19], a human-edited directory, and the Alexa Top 500 list. Since these two sets of domains are legitimate and do not contain fast-flux domains, they can be used as a begin set to instantiate the weights. At first, we performed two consecutive DNS lookups of all domains. This lookup process took the TTL of each domain into account: We waited TTL + 1 seconds between two lookups to make sure not to get a cached response from the nameserver in the second lookup. We repeated the lookup process several times.

In order to evaluate the detection performance of the proposed flux-score, we performed a 10-fold cross-validation on the corpus of labeled fast-flux and benign domains using different model parameters for finding the optimal hyperplane. The best model achieves an average detection accuracy of 99.98% with a standard deviation of 0.05%, thus almost no predictions on the testing data sets are incorrect. Regarding the weight vector w and bias b, the obtained assignments yield the following definition of the flux-score:

$$f(x) = 1.32 \cdot n_A + 18.54 \cdot n_{ASN} + 0 \cdot n_{NS} \quad (2)$$
$$\text{with } b = 142.38$$

Note, that the weight corresponding to n_{NS} is 0 and does not contribute to the detection of current FFSNs. Even though the flux-score is constructed from only two observed parameters, evading detection is difficult as the involved parameters n_A and n_{ASN} reflect essential properties of the underlying distributed structure of a FFSN.

The values of w_1, w_2 and w_3 as well as the threshold should be adjusted periodically since attackers could try to mimic CDNs in order to evade our metric, e.g., by sorting the IP addresses from their flux-agents according to IP address and then return only sequences of IP addresses that look like CDNs. We claim however that due to the two restrictions described above, it is hard for scammers to mimic *exactly* the behavior of a CDN. A fundamental difference between FFSNs and CDNs remains: A FFSN is built on top of compromised machines and the attacker has only limited influence on the availability, e.g., the user of the compromised machine can turn off the machine at arbitrary times. As part of future work, we want to examine how we can build a metric that automatically adapts to changes in FFSNs. This could for example implicitly include the fluxiness φ since φ for benign domains reaches its saturation limit pretty quickly comparing to fast-flux domains which have a growing fluxiness over time. In particular, benign domains with only one fixed IP have a constant φ $(= 1)$ from the very beginning of repeated DNS lookups. We would sacrifice our fast detection metric (only two DNS lookups), but could possibly also detect stealth FFSNs.

4 Empirical Measurements on Fast-Flux Service Networks

We now present results of studies on fast-flux domains and the underlying scam infrastructure. This is the first empirical study of the fast-flux phenomenon giving details about FFSNs we observed during a two-month period in July/August 2007. Most important, we demonstrate that some results of a previous study in this area [3] have changed, since scammers now operate differently because they adopted FFSNs and use this technique to host their scams. Even in the short period since the results of that study, there is already a change in tactic by the scammers.

4.1 Scam Hosting via Fast-Flux Service Networks

In this section, we focus on how FFSNs are used by spammers, e.g., for hosting websites that offer pharmaceutical products or replica watches. We study the domains found in spam e-mails (*spamvertized* domains). There are commonly both fast-flux and benign domains in this set: A spammer could host the online shop for his scam on a normal server or use a FFSN to have a more reliable hosting infrastructure which is hard to take down.

FFSNs Used by Spammers. Our study is based on a spam corpus from `http://untroubled.org/spam/`. The corpus contains 22,264 spam mails collected in August 2007 with the help of spam-traps, thus we can be sure that our corpus contains only spam. From all these mails, we were able to extract 7,389 unique domains. We performed two `dig` lookups on all these domains and computed the flux-score according to Equation 2. In total, we could identify 2,197 (29.7%) fast-flux domains in the corpus. Anderson et al. used in their study a spam corpus collected in November 2006, and they did not find any FFSNs [3]. They found that 6% of scams were hosted on multiple IP addresses, with one scam using 45. All the scams hosted on multiple IP addresses could be FFSNs, which were not identified as such.

The two `dig` lookups for all fast-flux domains identified in our spam corpus resulted in an observation of 1,737 unique IP addresses pointing to compromised machines running flux-agents. This demonstrates that FFSNs are a real threat and nowadays commonly used by attackers. By performing a reverse DNS lookup, we confirmed that the flux-agents are commonly located in DSL/dial-up ranges, thus presumably belong to inexperienced users.

The majority of the fast-flux domains (90.9%) consist of three domain-parts. For these, the third-level domain is usually a wildcard that acts as a CNAME (*canonical name*) for the second-level domain. To exclude wildcards, we only take the top- and second-level domain into account: In total, we can then identify 563 unique fast-flux domains. Only four top-level domains are targeted by attackers: `.com` was used 291 times (51.7%), `.cn` 245 times (43.5%), `.net` 25 times (4.4%), and `.org` twice (0.4%).

Similarity of Scam Pages. We also want to study how many scams are hosted via these FFSNs. Similar to a previous study [3], we want to explore the infrastructure used in these fraud schemes. We thus downloaded a snapshot of the webpage of each IP addresses returned in the `dig` lookup. The retrieved webpages comprise various dynamic content such as sessions numbers or randomized hostnames. These random strings, however, render analysis of content common to multiple webpages at the same IP address difficult. To tackle this issue we apply so called *string kernels*: A comparison method for strings, which is widely used in bioinformatics to assess similarity of DNA sequences [16, 26]. Compared to *image shingling* [3], a graphical method to find similar webpages, string kernels are more resilient to obfuscations: We found several webpages that simply changed the background image and such similar sites can not be identified via image shingling. Furthermore, string kernels enable comparison of documents with linear run-time complexity in the number of input bytes and yield performance rates up to 5,000 comparisons per second [23, 27]. Moreover, as our approach considers the original HTML documents, no further preprocessing or rendering of HTML content is required. Thus using a string kernel approach to detect similarity of scam pages is significantly faster than image shingling.

For our setup, we employ a string kernel that determines the similarity of two webpages by considering n-grams (substrings of n bytes) shared by both pages. Given webpages p_1, p_2 and an n-gram a shared by p_1 and p_2, we first define $\phi_a(p_1)$ and $\phi_a(p_2)$ as the number of occurrences of a in p_1 and p_2, respectively. The string kernel is then defined over the set A of all shared n-grams as

$$k(p_1, p_2) = \sum_{a \in A} \phi_a(p_1) \cdot \phi_a(p_2)$$

Note, that $k(p_1, p_2)$ corresponds to an inner-product in a vector space, whose dimensions are enumerated by all possible n-grams. Since $\phi_a(p_1)$ and $\phi_b(p_2)$ are natural numbers k is not bounded, so that we need to normalize it by considering a normalized variant $\hat{k}$

$$\hat{k}(p_1, p_2) = \frac{k(p_1, p_2)}{\sqrt{k(p_1, p_1) \cdot k(p_2, p_2)}}$$

The output of the kernel $\hat{k}$ is in the range 0 to 1, such that $\hat{k}(p_1, p_2) = 0$ implies that no shared n-grams exists and $\hat{k}(p_1, p_2) = 1$ indicates equality of p_1 and p_2. For all other cases $0 < \hat{k}(p_1, p_2) < 1$, the kernel $\hat{k}$ can be used as a measure of similarity between webpages.

Grouping of Webpages. Using string kernels, we can define an intuitive method for determining groups of similar webpages located at a given IP address. For each address we compute a matrix K, whose entries K_{ij} correspond to kernel values $\hat{k}(p_i, p_i)$ of the i-th and j-th webpage. We then define a similarity threshold $0 < t < 1$ and assign two webpages p_i and p_j to the same group if $k(p_i, p_j) > t$ holds. Computing these assignments for all pages is carried out by simply looping over the columns of the matrix K. Empirically we found a value of $k = 0.85$ to be a good threshold using the string kernel method for grouping webpages into scam hosts with the same content.

Fig. 6 shows the distribution of retrieved webpages per flux-agent. Please note that the x-axis is grouped into bins that grow quadratically. A little more than 50% of the flux-agents host just one webpage, but several pages per IP are not uncommon.

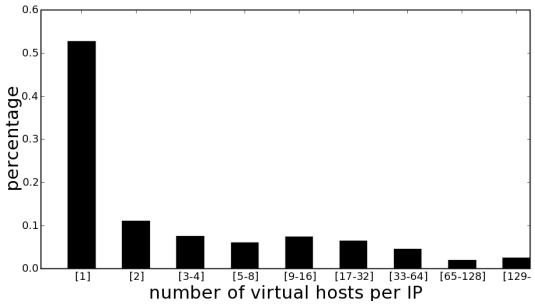

Figure 6: Distribution of virtual hosts per IP address per flux-agent

In Fig. 7, we depict the distribution of unique scams hosted on one particular IP after having applied the grouping algorithms. We see that commonly attackers just proxy one scam via one flux-agent, but in 16.3% of our observations, also multiple scams are proxied through one particular flux-agent. This is significantly less than in the study performed by Anderson et al. [3], who found that 38% of scams were hosted on machines hosting at least one other scam. This indicates that scammers can now have a broader distribution of their infrastructure due to FFSNs.

4.2 Long-Term Measurements on Fast-Flux Service Networks

To study the long-term characteristics of FFSNs, we observed several of them over a longer period of time: For 33 fast-flux domains found in spam e-mails, we performed a DNS lookup every 300 seconds over a period of seven weeks. In total, we observed 18,214 unique IP addresses during the measurement period between July 24 and September 10, 2007. This confirms that FFSNs are a real

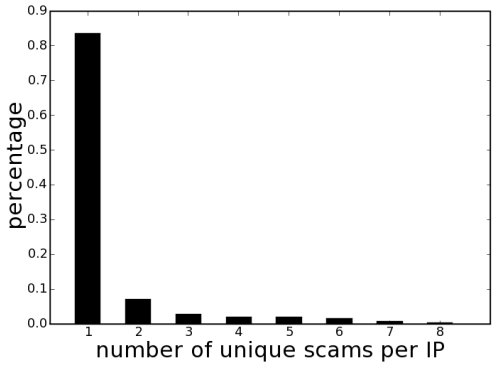

Figure 7: Distribution of unique scams per IP address per flux-agent

threat, with thousands of machines being compromised and abused. The monitored IPs belong to 818 unique ASNs and Table 2 lists the top eight ASNs we found. We see a wide distribution of flux-agents all over the (networked) world. Furthermore, the distribution follows a long-tail distribution, with 43.3% of all IPs contained in the top 10 ASNs.

This measurement does not take churn effects caused by DHCP into account[1]. However, we estimate that the percentage of churn caused by DHCP is rather small: In order to be a reliable flux-agent, the machine should be online for a longer time as otherwise it could cause downtime for the scam infrastructure. Thus an attacker will make sure to only include *stable* nodes, i.e., nodes that have a high uptime and constant IP address, into the pool of IP addresses served within FFSNs.

For each of the 33 FFSN domains, we examined the *diversity* of returned A records: Each time we observe a new A record, we assign an ascending ID to this IP address. This allows us to keep track of how often and when we have seen a particular IP. Fig. 8 plots the diversity of IPs for a period of 12.5 hours for two exemplary fast-flux domains. We see a wide variety of IPs returned in our DNS lookups and a steady increase of new fast-flux IPs we monitor (leading to an increase in the fluxiness φ). The slope of both graphs is different, indicating that different FFSNs have a different value for φ. Furthermore, this graph also highlights the dimension of flux-agents: within the short amount of time, more than 600 unique flux-agents were returned for the fast-flux domain in the lower part of the figure.

In contrast we also plot IP diversity over time for two benign domains in Fig. 9. Please note that the measurement period for benign domains is ten times more DNS lookups to show some effects in the plot. For the CDN domains, we observe only a small total number of unique IP addresses

[1]NAT is no problem since a flux-agent needs to be reachable in order to serve as content proxy.

1)	7132	(AT&T Internet Services, US)	2,677	2)	9304	(Hutchison Global, HK)	1,797
3)	4766	(Korea Telecom, KR)	590	4)	3320	(Deutsche Telekom, DE)	500
5)	8551	(Bezeqint Internet, IL)	445	6)	12322	(Proxad/Free ISP, FR)	418
7)	8402	(Corbina telecom, RU)	397	8)	1680	(NetVision Ltd., US)	361

Table 2: Top eight ASNs observed while monitoring 33 fast-flux domains over a period of seven weeks. The table includes the name and country of the AS, and the number of fast-flux IPs observed in this AS.

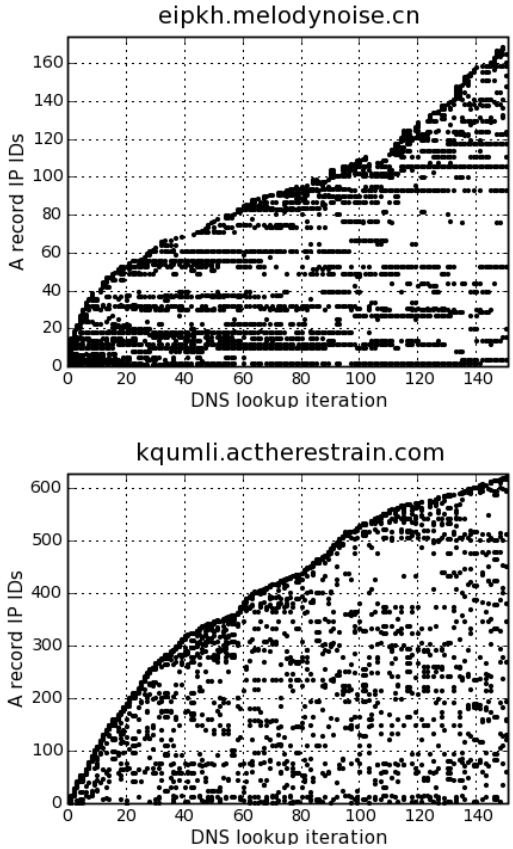

Figure 8: IP address diversity for two characteristic fast-flux domains

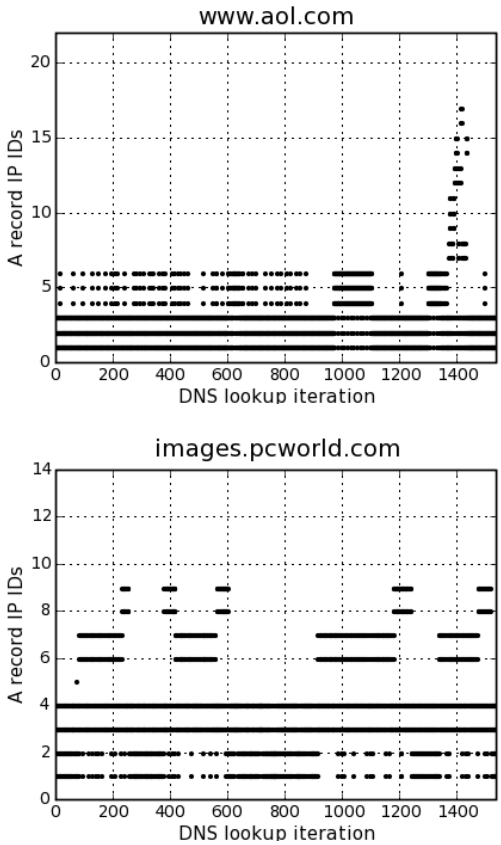

Figure 9: IP address diversity for two characteristic domains hosted via CDNs

returned and a clear pattern.

We found the fluxiness φ to be a reliable feature in case of many repeated DNS lookups: Even though it grows for both CDNs and fast-flux domains during the first DNS lookups, a saturation can be seen earlier for the CDNs and hence we can reliably decide fast-flux after repeated lookups by only considering the number of unique IP addresses observed during lookups, i.e., n_A.

To further study the long-term growth of n_A and n_{ASN}, we present in Fig. 10 the cumulative number of distinct A records and in Fig. 11 the cumulative number of ASNs observed for each of the 33 fast-flux domains during a period of more than 15 days. We see three different classes of growth in both figures. This means that different fast-flux domains have a characteristic distribution of flux-nodes. If two domains have a similar growth of the cumulative number of distinct A records or ASNs, this could indicate that both domains belong to the same FFSN: the nameservers return A records with similar characteristics. We plan to examine this in the future as a possible way to identify different domains belonging to the same control infrastructure. Furthermore, the two figures also show a declining growth over time. A longer measurement of the number of distinct A records or ASNs could be used to estimate the overall size

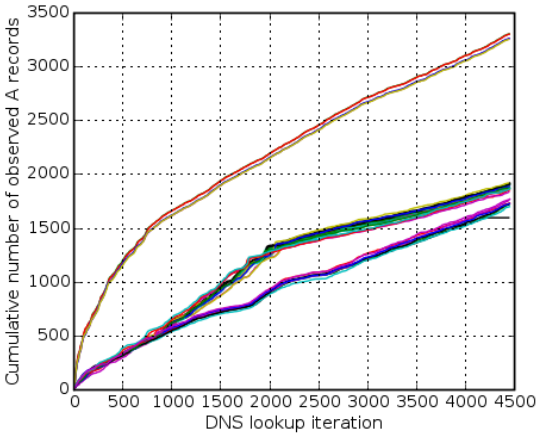

Figure 10: Cumulative number of distinct A records observed for 33 fast-flux domains.

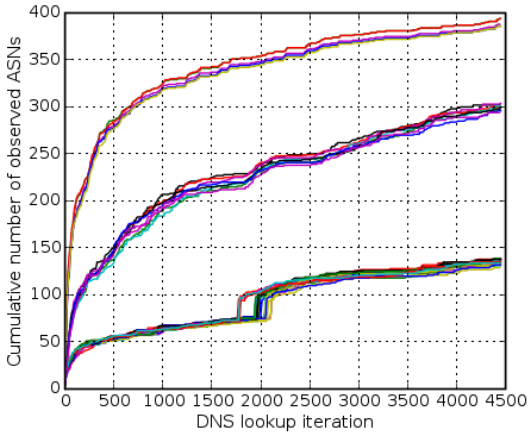

Figure 11: Cumulative number of distinct ASNs observed for 33 fast-flux domains.

of the pool of compromised machines used for a particular FFSN: Since the curves will eventually reach a saturation, this is an upper bound of the pool size. The initial strong growth in Fig. 11 also indicates that flux-agents are commonly spread over several ASs, confirming our weighting for the flux-score.

One goal of FFSNs is to provide a robust hosting infrastructure for cybercriminals. However, we found that FFSNs also need to deal with unavailability of the site, especially caused by the unavailability of the DNS server itself. From the 374,427 DNS queries during the measurement period, 16,474 (4.60%) failed. We also monitored 16 legitimate domains from the Alexa Top 500 to measure the reliability of benign domains. From 128,847 lookups against these domains, only 17 (0.01%) failed.

4.3 Fast-Flux Service Network Characteristics

In FFSNs, several flux-agents are used for proxying content provided by a single control node, as schematically shown in Fig. 5. We now want to study this infrastructure more in-depth. The individual agents are compromised machines, used by attackers to form a proxy network providing them with a robust hosting infrastructure. By sending different types of HTTP requests, we are able to obtain *service banners*, i.e., version and configuration information, from both the agents and the control node. The key idea is to send HTTP GET and TRACE requests and examine the differences in answers to both requests. While flux-agents and their corresponding control node usually have different service banners, we observed the same banner for control node behind different flux-agents. We also observed that all of the 300 probed flux-agents responded with the same banner to our requests. Probing the control node behind the same flux-agents, we found three different service banners, where one of these banners appeared 275 times among 300 probes. This indicates that there are at least three control node, presumably many more. Even though banners in general do not give trustworthy information on the server version, the empirical observation is consistent with our theory of mass-distributed, similar configured flux-agents and only a small number of unique control node that host the content. Furthermore, we also tested all Alexa Top 500 domains and the 5,803 domains randomly chosen from the *Open Directory Project* and we witnessed none of these servers having the particular `Server` header information we found when probing the flux-agents and the control node. Since we did not do any further investigation in this direction yet, we do not want to overrate these findings, but want to point them out as one idea on how to identify control nodes.

4.4 Other Abuses of Fast-Flux Service Networks

Besides using FFSNs to host scam sites related to spam, we also found several other illegal use cases for these networks. This is presumably due to the fact that FFSNs provide a robust infrastructure to host arbitrary content: They are not restricted to work with HTTP servers, but an attacker could also set up a fast-flux SMTP or fast-flux IRC server. In this section, we briefly provide information about two additional examples of how attackers use FFSNs as part of their infrastructure.

First, fast-flux networks are commonly used by phishing groups. *Rock phish* is a well-known phishing toolkit which allows an attacker to set up several phishing scams in parallel: The attacker installs the phishing kit on a webserver and different URL-paths lead to different phishing pages [18]. The actual domain belonging to these phishing pages commonly uses fast-flux. For example, the domain

regs26.com was used for phishing in September 2007. By performing a DNS lookup every time the TTL expired, we observed a total of 1,121 unique A records during a measurement period of four days.

Second, also botherders use FFSNs to host malicious content. The P2P bot *Storm Worm* [11], one of the most prevalent bots nowadays, uses fast-flux domains to host the actual bot binary. We monitored the domain tibeam.com for a period of four weeks in August 2007 and observed more than 50,000 unique IP addresses in the returned A records for this particular domain. This indicates that Storm Worm is a large botnet, since the pool of IP addresses served for the FFSNs is apparently large. Furthermore, monitoring the domain allows us to keep track of the botnet: since each compromised machine also runs a webserver, which hosts the actual bot binary, we can download the current binary from this host.

5 Mitigation of Fast-Flux Service Networks

In this section, we briefly review several strategies to mitigate the threat posed by FFSNs. Even though the client's view onto an FFSN is pretty limited (we can only monitor the flux-agents), we try to collect as much information as possible with the techniques outlined in the previous sections. Our metric helps us to automatically find fast-flux domains, which can be collected in a *domain blacklist*. First, such a blacklist can be used to stop a fast-flux domain with the help of collaboration from domain name registrars: A registrar has the authority to shut down a domain, thus effectively taking down the scam. An automated blacklist of fast-flux domains can quickly notify registrars about fraudulent domains. Second, an ISP can use such a blacklist to protect its clients from FFSNs by blackholing DNS requests for fast-flux domains. Third, the domain blacklist can be used for spam filtering: If an e-mail contains a fast-flux domain, it is most likely a spam mail. This technique could be used at the client- or server-side if slight delay is tolerable. Tracking of FFSNs by periodically performing DNS lookups for fast-flux domains can be used to build a list of IPs which most likely are compromised, which could be used in a similar way as the domain blacklist.

Similar to an anonymity system, a FFSN has one layer of redirection: A client can not directly observe the location of the control node, which hosts the actual content, but only the flux-agent. Ideas from this area can be adopted to identify the actual location of the control node: An ISP has the capability to monitor "from above" both the incoming and outgoing flows for a given machine and thus can monitor flows belonging to a flux-agent. If a flux-agent is located within the network of an ISP, the idea is to inject requests which are proxied from the agent to the control node (step 2 in Fig. 5). Together with the response to such requests, the location of the control node can be identified and this content-serving central host can then be taken down [28].

As FFSNs might be only the first step towards high available scams, we should also think of more general approaches on combating this kind of distributed, malicious infrastructure. One possibility would be to block certain incoming connection requests directed to dial-up ranges, e.g., to TCP port 80 or UDP port 53. The majority of dial-up users does not need to host servers, and such an approach would block many possibilities to abuse compromised clients. ISPs could change their policy to not allow any network services within mainly dynamic IP ranges by default. Still certain ports could be enabled by whitelisting if there is a need for network services for specific users.

6 Conclusion and Future Work

In this paper, we presented the first empirical study of FFSNs. For this study, we developed a metric that exploits the principles of FFSNs to derive an effective mechanisms for detecting new fast-flux domains in an automated way. Beside being straightforward to compute, we also showed that the method is accurate, i.e., we had very low false positive and false negative rates. As we are aware of the dynamics within fast-flux, we expect the need of further refinements in our method. Based on our empirical observations, we found other information, e.g., whois lookups and MX records, as promising features for an extended version of our flux-score.

Beside analyzing FFSN features in terms of detection and mitigation, we plan to work on statistical methods to estimate how many IP addresses are in a certain pool of one fast-flux domain. Adopting *capture-recapture* methods, which are applied in biology for measuring the number of members of a certain population [22], could be one way to obtain such an estimation. Similar methods were successfully applied by Weaver and Collins to measure the extent of phishing activity on the Internet [30] and we plan to study whether or not this can also be adopted for the area of fast-flux service networks.

Availability. To foster research in this area, the data collected during our study is available for research purposes. Interested readers are referred to http://pi1.informatik.uni-mannheim.de/fast-flux for information on how to obtain the data.

Acknowledgments. We would like to thank the anonymous reviewers for their helpful comments and our shepherd Fabian Monrose for his careful advice.

References

[1] Akamai Technologies. Akamai Content Distribution Network. http://www.akamai.com.

[2] Alexa, the Web Information Company. Global Top 500 Sites, September 2007. http://alexa.com/site/ds/top_sites?ts_mode=global.

[3] D. S. Anderson, C. Fleizach, S. Savage, and G. M. Voelker. Spamscatter: Characterizing internet scam hosting infrastructure. In *Proceedings of the 16th USENIX Security Symposium*, 2007.

[4] D. Barr. RFC 1912: Common DNS operational and configuration errors, February 1996. http://www.ietf.org/rfc/rfc1912.txt.

[5] A. Birolini. *Reliability Engineering: Theory and Practice*. Springer Verlag, 2004.

[6] P. Bradley and O. Mangasarian. Massive data discrimination via linear support vector machines. *Optimization Methods and Software*, 13:1–10, 2000.

[7] T. P. Brisco. RFC 1794: DNS support for load balancing, April 1995. http://www.ietf.org/rfc/rfc1794.txt.

[8] V. Cardellini, M. Colajanni, and P. S. Yu. Dynamic load balancing on web-server systems. *IEEE Internet Computing*, 3(3):28–39, 1999.

[9] F. Freiling, T. Holz, and G. Wicherski. Botnet tracking: Exploring a root-cause methodology to prevent distributed denial-of-service attacks. In *Proceedings of 10th European Symposium On Research In Computer Security (ESORICS'05)*, 2005.

[10] S. Gadde, J. S. Chase, and M. Rabinovich. Web caching and content distribution: a view from the interior. *Computer Communications*, 24(2):222–231, 2001.

[11] J. B. Grizzard, V. Sharma, C. Nunnery, B. B. Kang, and D. Dagon. Peer-to-peer botnets: Overview and case study. In *Proceedings of 1st USENIX Workshop on Hot Topics in Understanding Botnets*, 2007.

[12] Hewlett Packard Inc. NonStop home page: HP Integrity NonStop computing. Online: http://h20223.www2.hp.com/nonstopcomputing/cache/76385-0-0-225-121.aspx, Sept. 2007.

[13] Internet Software Consortium. dig: domain information groper, September 2007. http://www.isc.org/sw/bind/.

[14] E. D. Katz, M. Butler, and R. McGrath. A scalable HTTP server: the NCSA prototype. In *Selected papers of the first Conference on World-Wide Web*, pages 155–164. Elsevier Science Publishers B. V., 1994.

[15] B. Krishnamurthy, C. Wills, and Y. Zhang. On the use and performance of content distribution networks. In *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, pages 169–182, 2001.

[16] C. Leslie, E. Eskin, and W. Noble. The spectrum kernel: A string kernel for SVM protein classification. In *Proc. Pacific Symp. Biocomputing*, pages 564–575, 2002.

[17] J. Mirkovic and P. Reiher. A taxonomy of DDoS attack and DDoS defense mechanisms. *SIGCOMM Comput. Commun. Rev.*, 34(2):39–53, 2004.

[18] T. Moore and R. Clayton. An empirical analysis of the current state of phishing attack and defence. In *Proceedings of the Sixth Workshop on the Economics of Information Security*, 2007.

[19] Netscape Communications Corporation. ODP – open directory project. Online: http://dmoz.org, 2007.

[20] D. A. Patterson. A simple way to estimate the cost of downtime. In *Proceedings of the 16th USENIX System Administration Conference (LISA'02)*, 2002.

[21] D. A. Patterson, G. Gibson, and R. H. Katz. *A case for redundant arrays of inexpensive disks (RAID)*. ACM Press, New York, NY, USA, 1988.

[22] K. H. Pollock, J. D. Nichols, C. Brownie, and J. E. Hines. *Statistical Inference for Capture-recapture Experiments*. Wildlife Society, 1990.

[23] K. Rieck, P. Laskov, and K.-R. Müller. Efficient algorithms for similarity measures over sequential data: A look beyond kernels. In *Pattern Recognition, Proc. of 28th DAGM Symposium*, LNCS, pages 374–383, Sept. 2006.

[24] S. Saroiu, P. K. Gummadi, R. J. Dunn, S. D. Gribble, and H. M. Levy. An analysis of internet content delivery systems. In *Proceedings of 5th Symposium on Operating System Design and Implementation (OSDI)*, 2002.

[25] A. Shaikh, R. Tewari, and M. Agrawal. On the effectiveness of DNS-based server selection. In *Proceedings of IEEE INFOCOM 2001*, 2001.

[26] J. Shawe-Taylor and N. Cristianini. *Kernel methods for pattern analysis*. Cambridge University Press, 2004.

[27] S. Sonnenburg, G. Rätsch, and K. Rieck. Large scale learning with string kernels. In L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, editors, *Large Scale Kernel Machines*, pages 73–103. MIT Press, 2007.

[28] The Honeynet Project. Know Your Enemy: Fast-Flux Service Networks, July 2007. http://www.honeynet.org/papers/ff/.

[29] V. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, 1998.

[30] R. Weaver and M. Collins. Fishing for phishes: Applying capture-recapture methods to estimate phishing populations. In *Proceedings of 2nd APWG eCrime Researchers Summit*, 2007.