

# A Case Study on Asprox Infection Dynamics

Youngsang Shin<sup>1</sup>, Steven Myers<sup>2</sup>, and Minaxi Gupta<sup>1</sup>

<sup>1</sup> Computer Science Department,  
Indiana University,  
Bloomington, Indiana

{shiny,minaxi}@cs.indiana.edu

<sup>2</sup> School of Informatics,

Indiana University,  
Bloomington, Indiana

samyers@indiana.edu

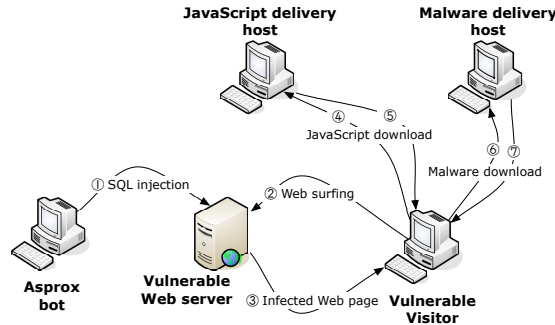
**Abstract.** The Asprox infection weaves a complex chain of dependencies involving bots that perform SQL injections on vulnerable web servers, and visitors whose machines get compromised simply by visiting infected websites. Using real-world data sets, we study Asprox bots, infected web servers, and the malicious infrastructure behind Asprox propagation. We find that the malware-propagation infrastructure in Asprox is aggressively provisioned to resist take-down efforts. This, combined with the easy availability of vulnerable user machines and web servers whose administrators are probably constrained in time and resources necessary to fix the problem, indicates that cleaning up Asprox infections is not going to be easy.

**Keywords:** Asprox, Malware, SQL Injection, Security.

## 1 Introduction

The Asprox botnet has been around since 2007. It was initially used exclusively for sending phishing emails. Around May 2008, a new update was pushed to Asprox bots in an attempt to grow the size of the botnet [1]. This update added an SQL injection vector. The updated bots could attack legitimate websites and inject scripts that redirected the browsers of visitors to these sites to infrastructure that cause the drive-by-download of malware. This malware is normally the Asprox botnet itself, but other payloads have been observed. Since the SQL injection vector was added, a significant number of web servers have been attacked and their unsuspecting visitor machines turned into Asprox bots [2] [3].

The Asprox botnet has a multi-step life-cycle, shown in Figure 1. The bots begin by using a search engine, such as Google, to find vulnerable web servers. Specifically, they search for servers that use `asp`, `aspx`, or `php` scripts to dynamically generate web pages from SQL databases. Next, the bots attempt an SQL injection attack on each server. If successful, the injection inserts malicious JavaScripts that invisibly redirects the browsers of visiting machines to malicious infrastructure. The infrastructure is also typically comprised of Asprox bots. Any visitor that browses an SQL-injected web server is at the risk of being infected



**Fig. 1.** Components of a typical Asprox infection. 1) A bot performs a successful SQL injection attack on a vulnerable web server. 2) A user visits the attacked server and views infected web pages, causing the download and execution of JavaScript code with `<script>` tags. 3-4) The (hidden) JavaScript invisibly directs the user’s browser to a malicious host by exploiting `<iframe>` tags. 5) JavaScript of the malicious host scans the visitor’s machine for vulnerabilities in the browser and OS, and redirects accordingly to potentially yet another final malicious site. 6-7) The final malicious site launches a drive-by-download that takes specific advantage of the vulnerability discovered in Step 5.

with Asprox malware, with infection ultimately depending on whether or not the visitor’s system is vulnerable to the attacks targeted by the drive-by-download. If the drive-by-download is successful, the typical payload includes—at the very least—the Asprox bot code. The life-cycle is complete, and ready to repeat.

As with all botnets, there are many types of harm and fraud that can be committed with Asprox. We have directly observed it being used to commit fraud through traditional phishing activities and fake virus protection scams. However, what makes Asprox interesting to study is both the multi-stage infection techniques it uses to spread, and the counter-countermeasures it uses to defend itself from take-down and detection attempts. Examples of such techniques include fast flux and JavaScript obfuscation.

**Goals of the Study:** In this paper, we analyze Asprox from three perspectives. First, we study the beginning of Asprox life-cycle by looking at the bots that were attempting to perform SQL injection on the web servers at Indiana University. Second, we study the extent to which the SQL injection attack is successful on a global population of web servers, and how long the infection persists by directly searching for infected servers on the Web. Finally, we study the defensive posture of the infrastructure responsible for scanning for vulnerabilities and delivering malicious content to machines whose browsers are redirected due to successful SQL injections. We also consider how it is provisioned to resist take-down attempts. Our study offers a unique perspective on the Asprox infection, a perspective that would not be possible through passive data collections techniques, such as honeypots.

**Key Observations:** We make several observations about the Asprox infection from our data: Though Asprox is a global phenomenon, over one third of the Asprox bots that were detected injecting SQL into vulnerable web servers were physically located in China. Yet only 5% of the SQL-injected web servers we found were located in China. Additionally, the the majority of the infrastructure that hosted the malicious JavaScripts and drive-by-downloads was located in the USA. This suggests that the botmasters are actively partitioning their infrastructure along technical, geographic or jurisdictional boundaries.

The Asprox bots exhibited activity patterns that strongly suggested that many of them were residential machines. Miscreants also seemed to rotate the bots, we hypothesize this is to prevent them from being sterilized by blacklisting of the bots. Our data did not contain any popular web servers (as defined by the Alexa data set [4]), indicating that those are either better secured or are quickly cleaned when SQL injections are found. A number of high-profile sites that garnered media attention when they were found to be infected suggest that it might be the latter case, more than the former. Unfortunately, we measured that over 1/4 of the infected web servers in our data set continued to host SQL-injected pages for over 100 days, and many of them are likely still hosting the injection! The remaining 3/4 of the sites were cleaned, but 60% of these took over a month to be cleaned. This clearly points to the need for either better awareness and/or resources for the operators of these servers. We note we cannot definitively say if the sites remain uncleaned for so long because the administrators are not aware of the injection, or if it is because they lack the tools to easily clean their databases.

Finally, we found that 58% of the hosts that delivered malicious JavaScripts were provisioned using *fast flux* to actively resist take-down efforts.

**Roadmap:** The remainder is organized as follows: Section 2 describes the data collection; Section 3 presents our analysis of Asprox infection dynamics; related work is discussed in Section 4; and Section 5 concludes the paper.

## 2 Data Collection and Overview

We collected three types of data for analysis. The first data set contains a network log of the Asprox bots that tried to launch numerous SQL injection attacks on various Indiana University web servers in August of 2008. The second data set contains the URLs of the infrastructure that delivered the malicious JavaScript. The last data set contains information gathered about infected web servers, and how quickly they were cleaned, if at all. Although the first two data sets play different roles in the Asprox infection process, we believe that they are in the same Asprox botnet. This is because even though we gathered the two data sets over different time periods and from different sources, we actually saw 8 common IP addresses in those two data sets. This is a small overlap, but it implies that infected machines can act as either SQL injection attackers, or as JavaScript/drive-by-download hosts. In the following subsections we describe them and their collection method in a detail.

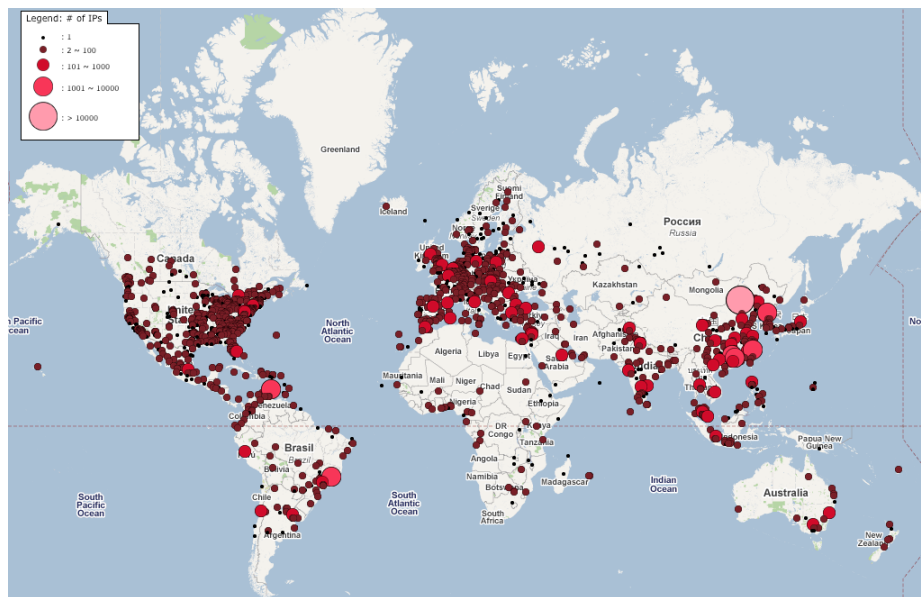
## 2.1 Data on Asprox Bots

The first data set contains the IP addresses and time stamps of the Asprox bots that launched SQL injection attacks on various Indiana University web servers in August 2008. The attacks were filtered via a router's firewall by looking for an Asprox SQL injection-specific string [5]. The attacks peaked in August 2008 causing massive amounts of unwanted traffic for the university. Table 1 shows the overview of this data set.

**Table 1.** Asprox bots and their targets

Collection period	8/9/2008 - 8/25/2008
Unique IP addresses of attacking bots	57,419
Autonomous systems attackers belonged to	1,847
Web servers targeted	581

Figure 2 depicts the geographical distribution of the IP addresses of the Asprox bots in our data. Though most of the bots belonged to North America, Europe, or East Asia, they cover much of the globe. *China had the largest number of Asprox bots; accounting for 36.68% of the attackers.* No other country accounted for more than 8% of the bots.



**Fig. 2.** Geographical distribution of Asprox bots (IP addresses) performing SQL injection attacks on Indiana University web servers (August 2008). Icon size is proportional to the number of attackers (IP addresses) in that location.

## 2.2 Data on JavaScript-Delivery Hosts

The second data set consists of URLs of the malicious infrastructure that delivered the malicious JavaScript and drive-by-downloads, and meta information about servers provisioning the domains related to such URLs. We refer to the hosts of these URLs as *JavaScript-delivery hosts* throughout this paper. These URLs were collected from [6], which has been tracking such URLs since the 5th of May, 2008.<sup>1</sup> On average, the site posted two new URLs each day. From October 2008 to the end of January 2009, we monitored all of the URLs that had previously appeared on [6], or that were newly added. (The attack continues but [6] stopped reporting new URLs on 11/26/08.)

**Table 2.** Overview of JavaScript-delivery URLs and host names

Collection period	10/26/2008 - 1/31/2009
Number of URLs	373
TLDs	13
<i>gTLDs</i>	5 (.com, .mobi, .net, .org, .name)
<i>ccTLDs</i>	8 (.ru, .cn, .jp, .cc, .tk, .kz, .eu, .me)
Unique host names	324
<i>with gTLDs</i>	151 (.com: 105, .name: 28, .mobi: 11, .net: 4, .org: 3)
<i>with ccTLDs</i>	173 (.ru: 127, .cn: 34, .jp: 4, .cc: 4, .tk: 1, .kz: 1, .eu: 1, .me: 1)

Table 2 breaks down the 373 URLs that were observed into different top level domains (TLDs), both generic (gTLD) and country-code TLD (ccTLD). The most popular TLDs for JavaScript-delivery hosts are `.com` and `.ru`.

To learn about how the JavaScript-delivery were provisioned, we collected metadata on the URLs. Specifically, DNS lookups were performed to obtain the IP address(es) of each host. We also periodically looked up the DNS servers used by these hosts in each level of the DNS hierarchy. For each of the IP addresses corresponding to the host name discovered in the A records<sup>2</sup> and each of their DNS servers (on all levels) we looked up their geographic location using the IP2Location software [7]. The lookups were performed every 15 minutes, until the host was no longer alive. We chose the 15-minute granularity to strike a balance between the validity of retrieved DNS records, which is typically no more than 5 minutes, and the overhead on our institution’s DNS resolver.

Table 3 provides an overview of the data obtained from our DNS resolutions and geolocation for the JavaScript-delivery hosts. Of the 324 hosts that delivered malicious JavaScripts, we could resolve only 55. This is because when we began data collection in October 2008, many of the older URLs posted on [6] were already inactive. The 55 resolved names yielded 2,214 unique IP addresses over

<sup>1</sup> We initially were collecting such data ourselves, but were forced to abandon this direction due to University requirement that we disconnect our infected bots.

<sup>2</sup> An A record is a DNS resource record used for storing an IP address associated with a domain name.

**Table 3.** Data collected through DNS resolutions of JavaScript-delivery hosts

(a) JavaScript-delivery hosts		(b) DNS servers for JavaScript-delivery hosts	
Resolved host names	55	Resolved DNS server names	619
IP addresses	2,214	IP addresses	147
Autonomous systems	308	Autonomous systems	67
BGP prefixes	898	BGP prefixes	115
Countries	64	Countries	11

our entire monitoring period for the DNS resolutions. *Interestingly, while more than 1/3 of the Asprox bots were located in China, 2/3 of the JavaScript-delivery hosts were located in the US.* Similar findings that say that the primary malware-serving infrastructure is located in the US have been reported in other botnet studies as well [8]. Our data set has fewer IP addresses than DNS server host names. This is because several DNS server host names resolve to the same IP address. Although a relatively small number of IP addresses are used to host DNS servers, they are distributed over different networks according to their ASN, BGP prefix, and country information, to make detection and take-down difficult.

### 2.3 Data on Infected Web Servers

Our third data set gathers information about web servers that were affected by the SQL injection attacks. To collect this data set we searched for web pages that contained the URLs pointing to the malicious JavaScript hosts listed in the second data set. We automated this search by using the Google AJAX Web Search API [9] and Yahoo! Web Search API [10]. Both APIs are limited: i) The Google API returns at most 64 results per query; and ii) The Yahoo! API limits each querier’s IP address to 1000 queries per day, but, unlike Google, there is no limit on the number of returned results per query. We invoked both of types of search APIs, using the malicious URLs as query keywords, and merged the returned results.

The result of these searches was a set of URLs from potentially-SQL injected<sup>3</sup> web servers around the world. For each URL, we first extracted the cached pages from both Google and Yahoo!, and the date the cache was made. Next, we visited the URLs. Each visit could *fail or succeed*. A *failure* result could be due to either the unavailability of the web server or the page. Specifically, a page may be

<sup>3</sup> Some web sites introduce Asprox and provide some of the URLs as an example, not as actual JavaScript embedding. Furthermore, Google or Yahoo search presents a few false positives that do not actually have URLs which we specify as keywords. Thus, we need to verify the infection of the web servers returned from Google or Yahoo search.

unavailable because the page was dynamically generated and is no longer valid. In contrast, *success* denotes that the requested page was returned.

If a page’s cache has the SQL injection, but our subsequent visit to the URL results in a failure, we cannot be sure if the server is still infected. Therefore, we classify such servers as *infected but unreachable*. For cases in which the visit succeeds, the retrieved page may be clean or infected (i.e., it does or does not respectively contain an SQL-injection of the given URL). When clean, we cannot be certain that the server or page was ever infected. To verify, we again turn to the cached page and only if the cached page confirms the presence of an infection do we consider the server to be infected. In this case we can deduce that the server was cleaned some time before our data collection, but after the page was cached. This case is labeled *infected, reachable, but undecidable* in Table 4. Finally, only in cases when the visit succeeds and the retrieved page contains the offending URL can we determine the duration for which the infection persisted with a high degree of accuracy. This case represents 56% of all the servers we examined. They are labeled *infected, reachable, and identifiable* in Table 4.

**Table 4.** Web servers infected by Asprox for a data collection period between 11/01/2008 - 01/31/2009

Class	# of Servers	% of Servers
Total number of infected web servers	8926	100%
Infected but unreachable	2751	30.82%
Infected, reachable, but undecidable	1141	12.78%
Infected, reachable, and identifiable	5034	56.40%

## 2.4 Limitations of Our Data Sets

There are a few limitations to our data sets. The first is our inability to comment on how many and which bots compromised which web servers. The Asprox bots in our first data set are not necessarily the ones that compromised the web servers we studied. Getting data from servers that knew they were under attack is difficult due to either an unwillingness to share the data, or a lack of data from the organization that ran these servers – they may not be collecting at the time of the attack, or they may be unaware that they were being attacked. Our second limitation is that our data on Asprox bots performing SQL injection attacks is only for the massive August 2008 attack on Indiana University web servers. This precludes studying attacker evolution. The third limitation stems from the inability to guarantee that there may have been other JavaScript-delivery hosts during our data collection period that were not reported by [6]. The fourth limitation is that we do not have data to understand how and when the JavaScript-delivery hosts redirected visitors to malware-delivery sites. Doing so would have required us to execute the malicious JavaScript and visit the malware-delivery infrastructure. Unfortunately, this limitation is a direct result of our University’s policy preventing us from knowingly installing malware and

then doing the traversals necessary to collect this data set. Finally, our data on infected web servers is limited by the Google and Yahoo! Web search APIs that only permitted us to collect a subset of the highest ranked search results. And those results were only in the English-dominated regions of the Web.<sup>4</sup> **In spite of all the limitations, we believe our data allows us to gain important insights into the Asprox infection.**

### 3 Analysis of Asprox Infection Dynamics

We now analyze each data set described in Section 2 to understand the dynamics of Asprox infections.

#### 3.1 Analysis of Asprox Bots

Figure 3 shows the numbers of unique SQL-injection attackers and the web servers they target on each day for which we have data (August 2009). One trend is clear: *The number of attacking bots is lesser on weekdays than weekends.* This comes as no surprise as this has been previously observed in other botnets, and is an artifact of the fact that many bots are residential machines which are likely to be available for longer on the weekends than weekdays. Correspondingly, the total number of attacks launched on the weekends are also higher than those on the weekdays. Further, the number of web servers attacked is also higher on the weekends than weekdays, as expected.

Next, we look at bot reuse. Figure 4 shows how many new attacking bots are observed each day and how they compare with those seen the day before or even prior. *Clearly, new bots are added to the pool as the week progresses, with peaks on Saturdays. Furthermore, some modest number of bots (up to 3000) are being reused. The change in the number of daily reused ones also follows that of new ones. This explains why more bots are observed on the weekends.*

To study in detail the differential between the behavior of Asprox bots on weekends versus weekdays, we picked a representative day from each. Figure 5 shows the number of attacks on these representative days broken down for every hour. (We normalized the time of attack in our data based on *attacker's time zone (based on IP address)* in this figure, and not the time zone of the attacked.) *On the weekday, the attacks peak at three times, 10am, 5pm, and 8:30pm.* These peaks roughly correspond to the start and end of the work day, and an early evening period after dinner. On weekends there is a more uniform distribution of attack times corroborating the lack of synchronization among users' schedules outside of the work week.

We also looked at the active lifetime of attacking bots. Figure 6a shows both how many days of the SQL injection attack at Indiana University a specific attacker was seen, and for how long a given Indiana University web server was

<sup>4</sup> We did not attempt to increase the probability that web pages based on other language were returned in our searches by explicitly choosing a language other than English as a search selection option in the Google or Yahoo search API.

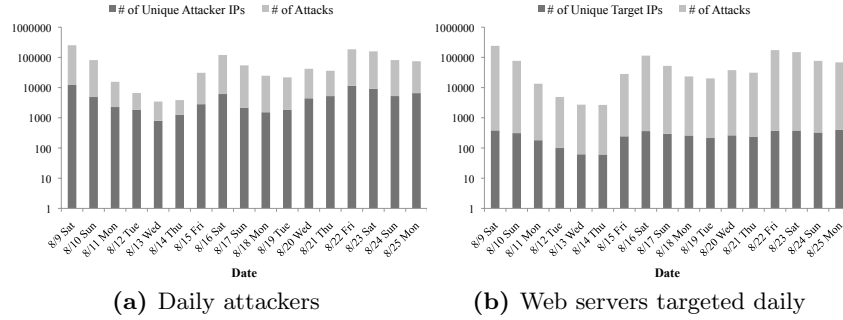


Fig. 3. Attackers' and targets' daily numbers

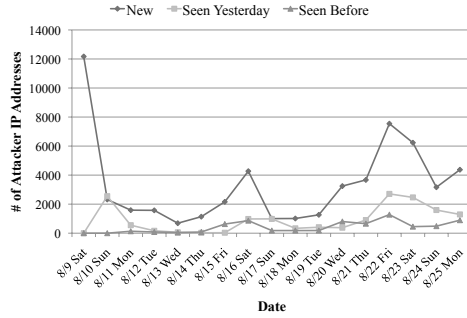


Fig. 4. New and old Asprox bots observed on a daily basis

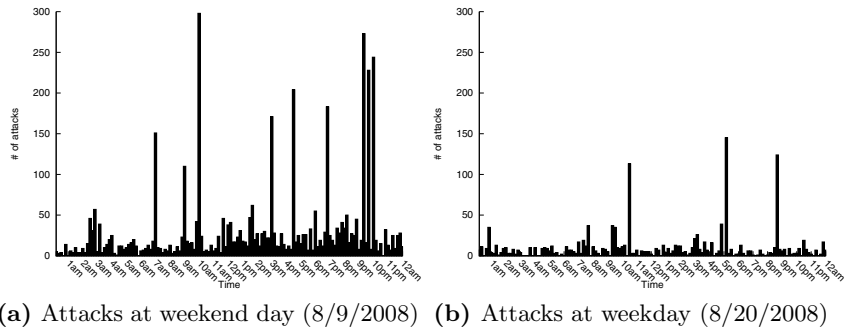
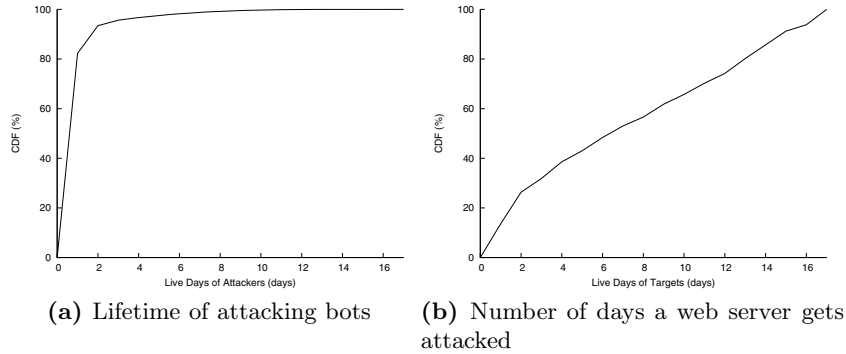
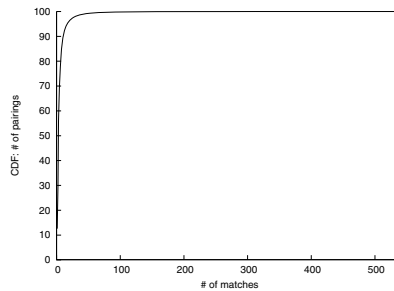


Fig. 5. Peak attack times on a weekend and a weekday. Times are normalized based on attacker's time zone.



**Fig. 6.** Attacker lifetime and the duration for which web servers face attacks



**Fig. 7.** Repeat attacks on Indiana University web servers by Asprox bots. The number of matches indicates how many times the same attacker attacks the same target. The number of pairings represents how many such a pair of an attacker and a target with the specific number of occurrences exists.

attacked. *Roughly 95% of attacking bots were observed for less than 2 days.* We hypothesize that this is to avoid any IP blacklisting. On the other hand, over 50% of web servers were continuously attacked for over 8 days. This seems to imply that the attackers are not systematic about avoiding repeat attacks on web servers – as they replace bots, the newer bots continue to attack servers that proved to be not vulnerable in the past.

Finally, we looked at the number of times an attacker repeatedly attacked any given target in Figure 7. *We find that 90% of the bots attacked the same web server about 10 times. In fact, in some cases one attacker hit the same target over 500 times.* Clearly, the Asprox bots are not synchronizing among themselves in choosing who they attack, nor keeping any individual state information on whom they have already attacked. This suggests that the bots independently search for web servers serving `asp`, `aspx`, and `php` pages and blindly attack the returned list. We cannot definitively say if the same machines that perform the

web-searches are those that launch the SQL injection attacks, or if one set of bots queries and passes lists off to another set of attackers. However, given the apparent lack of coordinate we hypothesize that they are the same. This would then explain the high frequency of attacks on some web servers. Additionally and/or alternately, since search engines are likely to index many pages from the same web server, the server gets targeted again and again.

### 3.2 Analysis of Infected Web Servers

An SQL injection attack is successful only if people visit the websites with infected pages. To see how often popular web servers get infected, we cross-referenced our list of infected web servers with the list of popular web servers reported by the Alexa [4]. There were no matches, confirming that the popular web servers are either better secured or quickly cleaned when successfully attacked. This certainly does not imply that the attackers do not target popular web servers. For example, there are confirmed attacks against Sony’s Playstation website [11]. Further, when looking at which web servers were targeted most often in our Asprox-bot data, we found that the two most popular web servers at Indiana University were those most frequently targeted in the attack. This, of course, also makes sense when we remember the targets are chosen based on the results of search engine queries, which generally put some emphasis in ordering based on popularity of the site.

Recall from Section 2.3, that of the 8,926 web servers we confirmed to have been infected at some point of time, 31% were unreachable. Another 13% were reachable and had been cleaned but not enough information was available to precisely lower-bound when they were cleaned up. For the rest of the 56% of the infected servers that were also reachable, we had sufficient data to estimate *lower bounds* on how long the servers stayed infected. Table 5 shows the TLDs these 56% web servers belonged to. As expected, `.com` was the TLD with most of the infected servers, claiming almost half of the positions. Other TLDs with more than a hundred infected web servers include: `.pl` (Poland), `.net`, `.org`, `.cn` (China), `.kr` (Korea) and `.uk` (United Kingdom). The infected servers belong to 6 other gTLDs and 87 other ccTLDs, confirming that the Asprox infection is truly a global epidemic.

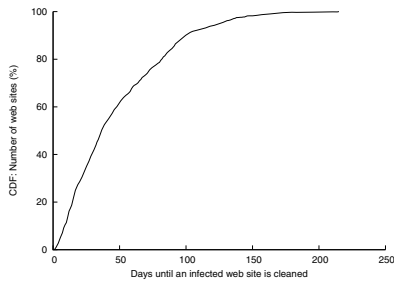
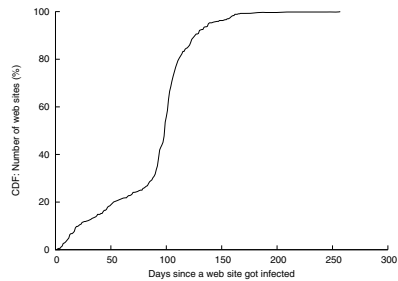
The 5,034 web servers reported in Table 5 were all victim to SQL injection because when we first retrieved their URLs found through Google and Yahoo! web searches, the URL inserted by SQL injection was still present in those pages. To determine how long the servers would remain SQL-injected, we both retrieved the same pages from Google and Yahoo! caches and retrieved them actively every day throughout our data collection process. When the page’s cache date was not available in the cache, which was the case for 54% of the servers, we considered only the infection duration from our active measurements. In either case, both are lower bounds for how long web servers stayed infected, and thus are conservative bounds.

Figure 8 shows the CDF of the servers that were cleaned up. Figure 9 shows the CDF of servers that still contained SQL-injections at the end of our data

**Table 5.** TLDs of infected web servers that were reachable and whose infection dates could be estimated

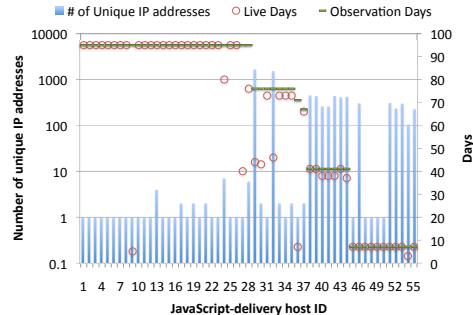
TLD	Number of web servers
.com	2307
.pl	341
.net	313
.org	294
.cn	242
.kr	201
.uk	125
Other gTLDs	105
Other ccTLDs	1070
No server name, just IP address	36
Total # of web servers	5034

collection. Overall, 77% of the servers were cleaned and the rest stayed infected. In fact, it took over a month for 60% of the web servers to be cleaned. Of those, 55% stayed infected for over 100 days! These observations reveal that the SQL injection component of Asprox is effective for the attackers and damaging to the visitors of these web servers. Because there are no usage statistics on any of the sites, we could not estimate how many browsers might have visited the sites over such a time period.

**Fig. 8.** Duration of infection for web servers whose infection was cleaned within our data collection**Fig. 9.** Status of web servers whose infection persisted till the end of our data collection

### 3.3 Analysis of JavaScript-Delivery Hosts

We now analyze the infrastructure that serves malicious JavaScripts (and potentially drive-by-downloads) to the unsuspecting visitors of infected web servers. Recall that our data contains the DNS resolutions for JavaScript-delivery hosts that were *live*, including the DNS servers and their IP addresses, and for each level of the DNS hierarchy.



**Fig. 10.** Number of unique IP addresses and live days for each JavaScript-delivery host. *Live days* is defined as the days when we succeed to do DNS resolution for a host. *Observation days* is the period for which we tried to do a DNS resolution for a host.

We focus on Asprox’s use of *fast flux* to provision the infrastructure used to deliver the malicious JavaScript. Fast flux is a recent technique used by attackers to keep their phishing and malware campaigns afloat for longer, and its goal is to prevent take-down and blacklisting. The key idea involved in fast flux is to have the attacker rapidly change the DNS bindings between the host name and its IP address. Two key indicators that fast flux is being used is that in the DNS resolutions of hosts are: 1) Host names resolve to a large number of IP addresses, generally scattered across many domains; and 2) each IP address has a short validity. The large number of IP addresses ensure availability under take-down attempts. The short validity ensures that any subsequent queries for the host name gives the attacker an opportunity to revise the list of IP addresses, should an IP address be taken down, blacklisted, or the supporting bot is shutoff. It is not known if IP addresses are spread across many administrative domains simply due to an artifact of a random subset of the botnet population, or if it results from an active attempt to prevent too many IP addresses under the same domains from coming down.

Fast flux comes in two flavors, which can be used independently or concurrently. The first is when the host itself fluxes. The second is when the DNS servers used to resolve the host name themselves flux. To distinguish the two cases we will term the former as fast flux and the latter as *DNS flux*. When both types of flux are used, we term the case as double flux. The Asprox botnet uses double flux.

Recall, as discussed in Sec. 2.2, that we collected data on the live JavaScript-delivery hosts, including information about their DNS servers. For each host observed, Figure 10 presents the number of unique IP addresses it resolved to through our observations, as well as the number of days each host was up. While phishing servers are often known to be taken down within a few days [12], the Asprox JavaScript-delivery hosts seem to survive for a surprisingly long time.

We note that hosts 1 through 28 were reported by [6] before we started collecting data. Thus, the fact that most of these hosts had only one IP address over the data collection period suggests that it may be in an artifact of data collection.

In particular, the botnet masters may have given up on using them, due to their detection. We do note that all of these hosts except one are currently blacklisted by Google, as we verified with a Firefox browser. In an attempt to validate the above hypothesis we verified the `whois` information for domains 1 through 28. It was returned for only 7 of them. Two of the seven seem to have been bought by a domain management company. Another two have been created and maintained by two organizations; one of them has even been removed from the blacklists. The remaining 3 hosts had very poor quality `whois` information, which is a common feature for attacker-controlled domains, and so these are likely to be still under the control of Asprox’s bot masters. One might question why the Asprox botnet masters would even want to maintain control of these domains, if they do not seem to be used, and they are blacklisted. We hypothesize that they are no longer in use precisely because they are blacklisted, but the masters keep control of the domain in case they are removed from the blacklists. It should of course be noted that any web server that has an SQL injection which points to an inactive host is safe for its legitimate users to visit. Hosts 29 through 55 in Figure 10 were part of the JavaScript-delivery infrastructure during our data collection period. 58% of these hosts appear to be actively fluxing. We examine this in more detail next.

To examine the details of fast flux, we use the three *fluxiness* metrics proposed by Holz et al in [13]:

$$F_A = N_A/n_{A\_average\_single} \quad (1)$$

$$F_{NS} = N_{NS}/n_{NS\_average\_single} \quad (2)$$

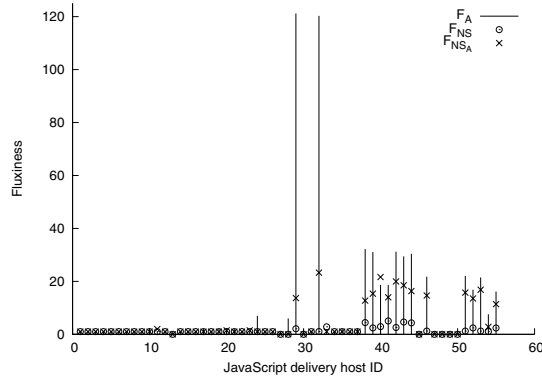
$$f_{NS_A} = N_{NS_A}/n_{NS_A\_average\_single} \quad (3)$$

$$F_{NS_A} = (\sum f_{NS_A})/N_{NS} \quad (4)$$

Here,  $F_A$  represents the degree of fast flux for each JavaScript-delivery host.  $N_A$  is the number of unique A records, that is, IP addresses returned in all DNS resolutions for a host. The value,  $n_{A\_average\_single}$  is the average number of A records returned for a single DNS resolution. [13] uses  $n_{single}$  which is the number of A records for a single lookup. However, as we observe the returned A records, the number are periodically changed. Thus, we use an average value for it instead. Host 29 and 32 give the highest value for it. Another fast fluxing hosts present a similar degree of fast flux. The others do not show fast flux at all. For double flux,  $F_{NS}$ , and  $F_{NS_A}$  describe the degree of DNS flux for the DNS servers of the JavaScript-delivery hosts.  $N_{NS}$  is the number of unique NS records, that is, DNS servers returned in all DNS resolutions for a host.  $n_{NS\_average\_single}$  is the average number of NS records returned for a single DNS resolution.  $N_{NS_A}$  is the number of unique A records, that is, IP addresses returned in all DNS resolutions for a DNS server.  $n_{NS_A\_average\_single}$  the average number of A records for a DNS server, which is returned for a single DNS resolution. Thus,  $f_{NS_A}$  is *fluxiness* for each DNS server.  $F_{NS_A}$  is the average of  $f_{NS_A}$ .  $F_{NS}$  shows that in the double fluxing hosts, the returned NS records for them are changed although its degree

is low.  $F_{NSA}$  shows that each DNS server also does fast flux and its degree is higher than  $F_{NS}$ .

Figure 11 shows all types of flux for each host. Although  $F_{NSA}$  seems to be comparable to  $F_A$  for each host except those for Host 29 and 32,  $N_{NS}$  is much smaller than  $n_{A\_average\_single}$ . *The small number of IP addresses for the DNS servers implies that taking the DNS servers down is a fruitful avenue to fight the Asprox malware-serving infrastructure.*

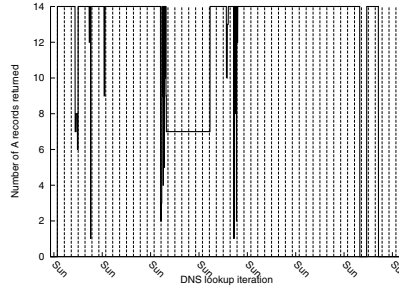


**Fig. 11.** Fluxiness:  $F_A$ ,  $F_{NS}$ , and  $F_{NSA}$

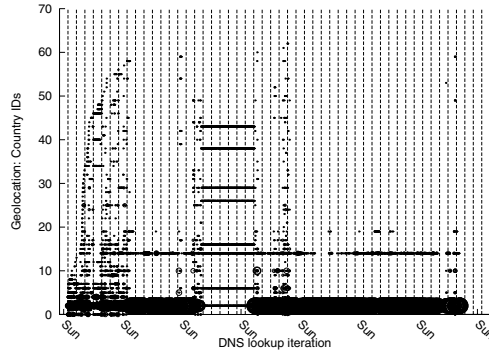
Hosts 29 and 32 are `www.81dns.ru` and `www.berjke.ru` exhibiting the highest degree of fast flux. They are active for over 40 days and resolve to 1669 and 1542 unique IP addresses respectively. We now examine them in detail. First, we see if they share IP addresses. Surprisingly, they shared 1397 (over 82%) IP addresses. Since these hosts are very similarly provisioned, we examine `www.berjke.ru` in more detail.

Figure 12 shows the number of A records returned by DNS resolution for the host name, `www.berjke.ru`. *On average, most of resolutions for `www.berjke.ru` have 14 A records.* Around 4 A records are changed each resolution with a standard deviation of 4. However, for most of the 4th week of observation they resolved into 7 fixed A records. We hypothesize that this may be the result of a bug on their part, which they fixed over the period of a week. In addition to seeing listed A records, we checked to ensure they are reachable by connecting to the IP address via port 80 (the JavaScript URL is over the HTTP protocol whose default port number is 80). On average, over 90% of IP addresses were reachable. Thus, they were highly available. Although modern web browsers understand round robin DNS and try another IP address when the chosen IP address is not reachable, this high availability makes the infection less detectable.

Figure 13 depicts the geolocational distribution of IP addresses returned in our DNS lookups for `www.berjke.ru`. *The IP addresses for `www.berjke.ru` is geographically spread throughout 60 countries, representing over 25% of country*



**Fig. 12.** Number of IP addresses for the DNS resolution of `www.berjke.ru`. Each vertical line indicates a separation of a day. DNS lookup iteration means one DNS resolution, repeated every 15 minutes.



**Fig. 13.** Change in geolocational distribution of IP addresses for `www.berjke.ru`. The size of each point is proportional to the number of IP addresses observed in the given country. Each vertical line indicates a separation of day. Each iteration means one DNS resolution and is repeated every 15 minutes.

*codes*. This indicates that the bots in Asprox botnet are geographically well distributed even though the number of its observed TLDs are small.

Figure 14 illustrates the geographical distribution of the IP addresses of the JavaScript-delivery hosts in our data. Unlike the geographical distribution of Asprox bots in Figure 2, most of the bots belonged only to North America or Europe. While China had only 2.67% of the JavaScript-delivery hosts, the United States had the largest number, which was 65.90% of them. Potentially related, at one point during our observations we found that the malicious JavaScript-delivery hosts would check to see if the default language on the visiting browser was Chinese, and not attempt to infect the browser if it was. No other country accounted for more than 4% of the hosts.



**Fig. 14.** Geographical distribution of IP addresses of JavaScript-delivery hosts. The size of icons for each point is proportional to the number of IP addresses for that location.

## 4 Related Work

Various case studies have been done to understand worms. These studies share a flavor similar to ours. A recent such study traced the Blaster worm for an extensive period and showed that it persisted in spite of significant mitigation effort [14]. Botnets have been studied extensively, particularly in the context of spam and phishing [15] [8] [16] [17] [18]. The Asprox botnet shares many of the features reported by these studies, particularly as they relate to the use of residential machines. Recently, Brown presented the anatomy of the Asprox in [19]. It presents Asprox’s history and comprehensive infection process, and infected machine’s behaviors with various examples. It also introduces the case where Asprox botnet is used by Rock Phish hosts [20]. However, it does not investigate and quantify the detail infrastructural provisioning for Asprox botnet. Furthermore, it does not trace the web servers infected by Asprox SQL injection attack.

SQL injection is central to the Asprox infection discussed in this paper. It is a well known attack on web servers with back-end databases. Consequently, various approaches have been proposed for detection and prevention of SQL injection. They can be categorized into three types [21]: coding practices with defensive mechanisms, vulnerability detection by static analysis, and defense techniques preventing vulnerabilities as well as detecting them. The defensive techniques transform programs to prevent SQL injection attack so that the

programmers do not have to validate inputs. Furthermore, [21] proposes a technique to dynamically infer programmer intentions to overcome the limitation of static analysis, and transform applications based on the conjecture. If implemented, these techniques can prevent SQL injection, without which Asprox would fail to be as effective without a significant evolution.

Fast flux is an important aspect of the Asprox botnets. The use of this technique is the latest trend among botnets and several research works have studied it well. The HoneyNet Project and Research Alliance wrote a whitepaper on fast flux [22]. While the paper does not provide a model which can be used to identify fast flux, it provides several pieces of valuable information including two real-world examples of DNS resolutions for fast flux host names, and a case study of the activities of an infected system. Nazario *et al.* [23] investigated the behaviors of botnets behind fast flux. They found that 80% of fast flux domains were registered at least a month before actual use. They also found a long lifetime for fast flux domains, a median of 18.5 days after the domain has become actively used, with several lasting over 50 days. Holz *et al.* [13] examined fast flux networks through resolutions of domains contained in spam mails. Toward the same goal, Passerini *et al.* [24] make use a more extensive number of features. They find that 8-30% of spam domains use fast flux today. We simply use these works to study fast flux among hosts serving malicious JavaScripts in our data.

## 5 Concluding Remarks

The Asprox botnet continues to grow and infect web servers around the world. With extensive use of fast flux, it is well provisioned to resist take-down attempts. Honey pots, while useful for detecting individual phases of the attack, are insufficient to understand the attack in its entirety or to detect changes or modifications to the final vulnerabilities used to attack users machines, or the malware payload delivered. Therefore, an active monitoring—and thus more costly—approach will be necessary to monitor changes to this botnet. Due to the use of JavaScript obfuscation, multi-layer fast flux, and redirects, a take-down of the JavaScript-delivery hosts and malware-delivery hosts seems unlikely to succeed, unless better mitigation techniques are developed in the community for specifically these problems. Thus, there is an urgent need to ensure that users use anti-virus software and keep their operating systems patched.

While there are a number of web servers that have fallen prey to the SQL injection attack, halting this portion of the life-cycle seems unlikely as well. The ability to patch a large number of improper SQL-accessing scripts to perform proper input validation or use SQL bind parameters on different web servers can only be effectively done through a long education cycle that is unlikely to yield results any time soon. Positively and as expected, we did not find any of the popular sites injected with Asprox. Note that this is not due to a lack of attacks, but rather because these servers are secured better and/or restored quickly. Unfortunately, this is not true of a large portion of the Web, for many web servers are small, and are unlikely to be updated (or fixed) frequently or in a timely manner.

The most vulnerable part of the Asprox life-cycle that we could discern was the specific URLs that are injected as part of the SQL injection segment of the life-cycle. These URLs point to the JavaScript-delivery hosts and can be blacklisted, if identified quickly. In fact, Google is currently blacklisting many of the them already, so Firefox and other modern browsers that use their blacklisting services, are protected if the blacklists are properly updated.

## Acknowledgments

We would like to thank Matthew Rainey and Andrew Korty for providing data on Asprox bots that attacked Indiana University web servers. We also thank Bob Cameron of the FBI for discussions on the Asprox botnet, and for URLs to cross-reference against our own. IP2Location [7] provided their geolocation software for our use, for which we are grateful. Finally, we would like to thank Rob Henderson for his help in guarding our data collection when we were accidentally mistaken for being miscreants ourselves.

## References

1. Stewart, J.: Danmec/Asprox SQL Injection Attack Tool Analysis, <http://www.secureworks.com/research/threats/danmecasprox>
2. The Times: Asprox computer virus infects key government and consumer websites, [http://technology.timesonline.co.uk/tol/news/tech\\_and\\_web/the\\_web/article4381034.ece](http://technology.timesonline.co.uk/tol/news/tech_and_web/the_web/article4381034.ece)
3. CyberInsecure.com: Asprox botnet mass attack hits governmental, healthcare, and top business websites, <http://cyberinsecure.com/asprox-botnet-mass-attack-hits-governmental-healthcare-and-top-business-websites/>
4. Amazon.com, Inc.: Alexa web information service, AWIS (2008), <http://aws.amazon.com/awis>
5. Cisco News: ASPROX SQL Injection Attacks - Block them using a Cisco router (July 2008), <http://cisco-news.co.uk/2008/07/09/asprox-sql-injection-attacks-block-them-using-a-cisco-router>
6. Zino, M.: ASCII Encoded/Binary string automated SQL injection attack, <http://www.bloombit.com/Articles/2008/05/ASCII-Encoded-Binary-String-Automated-SQL-Injection.aspx>
7. Hexasoft Development Sdn. Bhd.: IP2Location geolocation service (February 2008), <http://www.ip2location.com/>
8. Anderson, D.S., Fleizach, C., Savage, S., Voelker, G.M.: Spamsscatter: Characterizing internet scam hosting infrastructure. In: USENIX Security (2007)
9. Google: Google AJAX Search API, <http://code.google.com/apis/ajaxsearch/>
10. Yahoo: Web Search APIs from Yahoo! Search, <http://developer.yahoo.com/search/web/>
11. Danchev, D.: Sony PlayStation's site SQL injected, redirecting to rogue security software (July 2008), <http://blogs.zdnet.com/security/?p=1394>

12. McGrath, D.K., Gupta, M.: Behind Phishing: An Examination of the Phisher Modi Operandi. In: USENIX Workshop on Large-Scale Exploits and Emergent Threats (2008)
13. Holz, T., Gorecki, C., Rieck, K., Freiling, F.C.: Measuring and Detecting Fast-Flux Service Networks. In: NDSS (2008)
14. Bailey, M., Cooke, E., Jahanian, F., Watson, D.: The Blaster Worm: Then and Now. In: IEEE Security and Privacy (2005)
15. Xie, Y., Yu, F., Achan, K., Panigrahy, R., Hulten, G., Osipkov, I.: Spamming Botnets: Signatures and Characteristics. In: ACM SIGCOMM (2008)
16. Rajab, M.A., Zarfoss, J., Monrose, F., Terzis, A.: A Multifaced Approach to Understanding the Botnet Phenomenon. In: ACM IMC (2006)
17. Zhuang, L., Dunagan, J., Simon, D.R., Wang, H.J., Tygar, J.D.: Characterizing Botnets from Email Spam Records. In: USENIX Workshop on Large-Scale Exploits and Emergent Threats (2008)
18. Holz, T., Steiner, M., Dahl, F., Biersack, E., Freiling, F.: Measurements and Mitigation of Peer-to-Peer-based botnets: A Case Study on Storm Worm. In: USENIX Workshop on Large-Scale Exploits and Emergent Threats, LEET (2008)
19. Brown, D.: Anatomy of the Asprox Bonet. ToorCon X - San Diego (September 2008)
20. McMillan, R.: 'Rock Phish' blamed for surge in phishing (December 2006), [http://www.infoworld.com/article/06/12/12/HNrockphish\\_1.html](http://www.infoworld.com/article/06/12/12/HNrockphish_1.html)
21. Bandhakavi, S., Bisht, P., Madhusudan, P., Venkatakrishnan, V.N.: CANDID: Preventing SQL Injection Attacks using Dynamic Candidate Evaluations. In: CCS (2007)
22. The HoneyNet Project: Know Your Enemy: Fast-Flux Service Networks (July 2007), <http://www.honeynet.org/papers/ff/>
23. Nazario, J., Holz, T.: As the net churns: Fast-flux botnet observations. In: International Conference on Malicious and Unwanted Software, MALWARE (2008)
24. Passerini, E., Paleari, R., Martignoni, L., Bruschi, D.: Fluxor: detecting and monitoring fast-flux service networks. In: Zamboni, D. (ed.) DIMVA 2008. LNCS, vol. 5137, pp. 186–206. Springer, Heidelberg (2008)