STRATEGIES FOR MONITORING FAKE AV DISTRIBUTION NETWORKS

Onur Komili, Kyle Zeeuwen
Sophos, Vancouver, Canada
Email {onur.komili, kyle.zeeuwen}@sophos.com

Matei Ripeanu, Konstantin Beznosov
University of British Columbia, Vancouver, Canada
Email {matei, beznosov}@ece.ubc.ca

ABSTRACT

We perform a study of Fake AV networks advertised via search engine optimization. We use a high interaction fetcher to repeatedly evaluate the networks by querying landing pages that redirect to Fake AV distribution sites. We identify several distinct Fake AV distribution networks, and we show that each network exhibits distinct updating behaviours. We propose optimizations for crawlers that explore Fake AV networks to leverage the strong fan-in property of these networks and, where possible, the periodic update behaviour of the network elements. We evaluate these optimizations and show that they can be used to drastically reduce the number of visits to the network, which in turn reduces the likelihood of being blacklisted.

1. INTRODUCTION

Fake anti-virus (Fake AV) attacks are a relatively recent social engineering scam. In the past few years these attacks have increased in prevalence [1] and profitability [2]. The scam is fairly simple: it deceives the user into believing that their computer has been compromised, and then offers an anti-virus solution for a fee. There are several reasons why the Fake AV scam is a tactic of choice for malware distributors: first, it relies on social engineering, thus affecting both patched and unpatched users. Second, the result of a successful attack is money, as opposed to something that can potentially be exchanged for money (e.g. a rare *World of Warcraft* [3] sword).

Monitoring Fake AV is challenging for several reasons: distributors use sophisticated malware distribution networks (MDNs) that rapidly change in composition and use numerous anti-crawler techniques including IP blacklisting. This presents a unique problem to security vendors: the MDNs update frequently so they must constantly be monitored to obtain up-to-date information that will help protect customers. However, repeated evaluations of the network can lead to blacklisting which prevents the security vendor from monitoring the network.

The purpose of this work is to explore solutions to overcome these challenges. In particular, we wish to maintain effective monitoring of Fake AV MDNs while avoiding blacklisting for as long as possible. To this end we study Fake AV networks by repeatedly evaluating the infected landing pages using a high interaction fetcher (HIF) that makes requests from multiple pools of IP addresses. We identify means to group Fake AV networks into distinct MDNs using pattern matching on URLs, and show that each MDN exhibits distinct update behaviours.

After studying the update behaviour of the MDNs, we propose optimizations to the re-evaluation logic that controls the re-evaluation frequency for each MDN. These optimizations provide a drastic reduction in the volume of fetches required to maintain a fixed level of monitoring. In turn this reduces the likelihood of blacklisting (which we observed on several occasions during our experiments).

2. BACKGROUND

Fake AV is one of the top malware threats today, in part due to the profit it generates for malware authors. Stone-Gross et al. [2] showed that three particular Fake AV MDNs generated a combined revenue of more than $130 million dollars annually.

This section presents the anatomy of Fake AV malware distribution networks (MDNs) and the challenges researchers face when studying them.

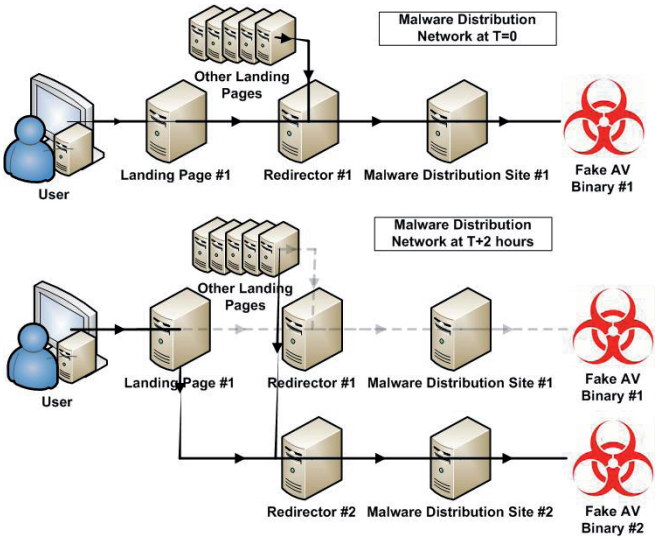

Figure 1: An example of how a typical Fake AV MDN changes over time.

Fake AV MDNs have evolved in several ways to evade the efforts of security researchers. A standard Fake AV attack starts when a user visits a landing page (e.g. one that appears as a search engine result). Landing pages are normally hosted on compromised but otherwise legitimate websites, typically running vulnerable content management software such as *Joomla* or *Wordpress*. Once they gain access to such a site, Fake AV distributors upload a malicious script (typically written in PHP) to the server that generates content to boost the ranking of the compromised pages. The generated content is usually related to trending topics [4] that users are likely to search for.

Our previous study [5] of search engine optimization (SEO) poisoning kits showed that these scripts use the HTTP user agent to differentiate crawlers from users. Additional checks are made on the HTTP referrer to confirm that the user came from an expected source (e.g. a search engine or a social networking site). We speculate that this is done to cloak the malicious behaviour from anti-malware researchers that may visit the site directly. If a user passes these checks, the script redirects the user into the MDN via a server side or client side redirect.

The use of client side redirection complicates the task of automatically harvesting Fake AV. Following a server side

redirection is a simple task. However, following client side redirects requires interpreting JavaScript. This is further complicated by the use of complex obfuscation and other anti-crawler JavaScript techniques [6]. Using a real web browser provides a solution to this problem; if a browser cannot interpret the content then the attack has failed, so in all cases a browser should be able to interpret the content rendered by an MDN.

When a user arrives at the final destination, they are presented with the standard 'Windows XP My Computer' page. Recently, variants of this page have appeared to match the OS and version of the user, for example a 'Windows 7 My Computer' screen [7], *Google*'s Safebrowsing warning page for *Firefox* users [8], *OSX Mac Finder* for *Mac* users [9], the *Microsoft Windows Update* page as shown in *Internet Explorer*, and numerous variations of the 'Windows XP My Computer' page. Regardless of the variant, the user is prompted to download and install a binary. To automate this process, some form of simulated human-computer interaction (HCI) is required; something needs to push the button. With the emergence of multiple looks and feel comes the need for more sophisticated HCI.

An additional challenge to efficiently monitor Fake AV MDNs comes from their dynamic nature: the *malware delivery tree* [6] (i.e. the path of redirections from a landing page to the executable download) and the binaries themselves are constantly changing. A study by Rajab et al. [1] showed that the median lifetime of a Fake AV domain decreased from 10 hours in September 2009 to under one hour in January 2010. Not only are the lifetimes of Fake AV domains dropping, but so too are the intermediary traffic direction systems (TDSs).

TDSs are used to direct traffic to malicious sites based on conditions such as the browser type, operating system, referrer, and so on. They are frequently found on bulk subdomain service providers (e.g. co.cc, cz.cc, co.be, co.tv) and have become increasingly popular over the past couple of years. Most of these offer free or low-cost subdomains – sometimes as little as $0.07–$0.10 for each domain when bought in bulk. At that price it is feasible for a malware distributor to rapidly throw away subdomains to circumvent the URL blacklisting and sample collection efforts of AV vendors. While not all Fake AV MDNs use these services, many do due to the reluctance of many AV vendors to block these services.

Fake AV MDNs make frequent updates to the malicious executable, in some cases on every request (i.e. server-side polymorphism). This requires frequent re-evaluation of the MDN to ensure that the malware is still detected. Unfortunately, simply re-fetching the repository URL is ineffective for several reasons. The first issue is that many binaries are served through a one-time or a time-sensitive URL that will deliver an executable only for a short window of time and then become inactive. The second issue is that the repositories are frequently changed, and so re-fetching a repository may result in downloading malware that is no longer being served to users. Finally, the lifetime of the repository domains themselves is often short, sometimes only a few hours. For the reasons above, it is clear that we need to start at the landing page and uncover the current malware delivery tree in order to guarantee a successful fetch of the latest version of malware.

We must also account for the unpredictable lifetime of landing pages. Landing pages are almost always innocent victims, and often (though not often enough), the landing pages themselves are cleaned up once the site administrator has become aware of the infection. This means we need to constantly find new landing pages or risk losing track of the MDN. We do this through a number of methods, including using *Sophos* customer feedback data and using the *Google* API [10].

A final, particularly challenging issue is that of blacklisting. We need to be careful not to expose our fetching infrastructure to the Fake AV distributors to the point where they can identify and blacklist it. Once a research client has been identified, the blacklisting response can take several forms. The simplest is a denial response where the malware is not delivered or the redirect is not issued. More devious responses include providing a static or non-malicious binary file.

These considerations lead us to conclude that the volume of resources required to monitor Fake AV is significant. We need to find heuristics that minimize client exposure (the number of requests) while maximizing the number of malware repositories and binaries discovered. This is the focus of the remainder of this paper.

3. DESIGN

The system we use is an in-house research tool named Tachyon Detection Grid (TDG)[1] [12, 13] (Figure 2). The system executes customizable experiments aimed at proving specific hypotheses by making HTTP requests at precise patterns using fetchers distributed across multiple IP pools. Our experiments consume lists of SEO poisoned landing pages and fetch them repeatedly using a high interaction fetcher (HIF). The system outputs a series of packet captures (PCAP) which are processed offline to draw conclusions.

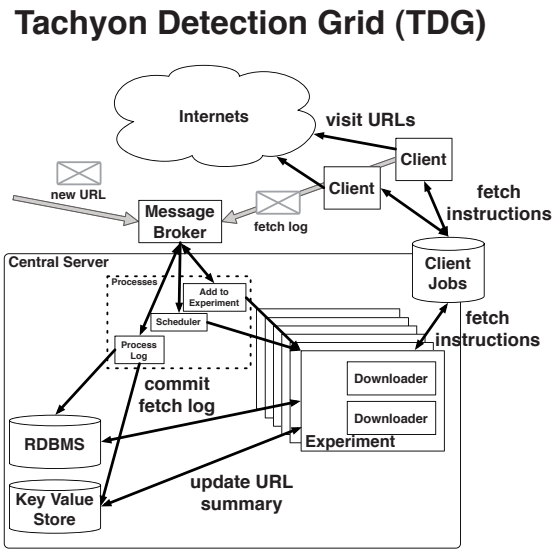

Figure 2: New URLs are added to the system and added to one or more experiments. The experiments delegate the role of fetching to the downloaders. The downloaders send command to the clients, which execute the HTTP requests. New fetch results are sent back to the central sever, where the experiment is invoked to process the new fetch.

[1] In *StarTrek: The Next Generation*, the Federation deploys a Tachyon Detection Grid (TDG) to detect cloaked Romulan vessels [11].

3.1 Landing page URL sources

The primary source of landing pages for our experiments are the confirmed SEO landing pages seen by *Sophos* customers. These enter *SophosLabs* through a feedback mechanism in our products that provides a detection name as well as the URL that triggered the detection. After some initial experiments with the *Google* API, we decided to rely solely on customer feedback for this research, as this feed contained far less noise than the feed from the *Google* API.

3.2 High interaction fetcher

We combined a number of technologies to create a fairly simple high interaction fetcher. With the constant need to start with a clean slate due to the possibility of infection, we chose to use *Virtual Box* with a snapshot of a *Windows XP* image. For our browser, we chose *Firefox*. This decision was motivated by the ease of modifying the initial HTTP referrer using a *Firefox* plug-in. A consequence of this decision is that we were not able to study exploits that target *IE* specifically. However, as our study focuses on Fake AV spread by social engineering, this is not an issue.

The landing pages check that the HTTP referrer is set to a search engine before redirecting. We were able to spoof this header using a *Firefox* add-on called RefControl [14]. In order to automate the human interaction component of the social engineering attack – the 'clicking of the buttons' – we use a tool called Sikuli [15]. Sikuli is written in Jython and uses image recognition capabilities to recognize when certain dialogs or links need to be clicked on the screen. We have a snapshot of the *Virtual Box* in a booted and ready state with our Sikuli script running in the background and actively looking for things to click.

3.3 IP address pools

In order to mask our infrastructure to reduce the likelihood of blacklisting, we need to be able to make requests from many different IP addresses. We use the *PlanetLab* network [16], a global research network with nodes spread across the world. We used roughly 75 *PlanetLab* nodes to proxy our traffic.

3.4 Post analysis

For post-analysis, we used an instrumented version of jsunpack [17] to parse the captured PCAP files and generate a structured JSON object representing the malware delivery tree. Details include fetch time, a full list of URLs visited and the SHA1 key of the content, the *Sophos* detection for each file, the method of redirection between pages, DNS information for each domain, and many other details.

4. RESULTS

We present results from analysis of 573 landing page URLs evaluated over a one-month period, which resulted in a total of 335 Fake AV repositories and 473 unique binary files. We identify specific MDNs that each have distinct characteristics. This section focuses on three main topics: first, it describes our solution to identify each Fake AV MDN; second, it presents the behavioural differences between the different families; and finally, it presents optimizations that reduce the number of MDN re-evaluations and, consequently, the likelihood of blacklisting.

4.1 Identifying the distinct malware distribution networks

The first step was to identify if there were any differences between all the various SEO poisoned landing pages. When investigating the landing pages, we found that there was no single particular exploit being used to infect all sites. Looking at the websites themselves, as best we could tell, the majority of the sites were infected through vulnerable versions of *Wordpress*, *osCommerce* and *Joomla*. Unfortunately, just looking at the URLs of the landing pages does not provide enough information to identify an MDN, since most use some randomly named PHP script, followed by query parameters that include the poisoned terms and in some cases a page number.

The next step was to visit the poisoned pages. At this point a number of differences became clear: the injected HTML was distinct for each MDN, as were the characteristics of the malware delivery tree. In all cases the initial redirection was done using JavaScript, however in the subsequent steps there was variation. One of the MDNs linked directly to the Fake AV repository page, while the remaining three linked to bulk subdomain service sites, which act as intermediary nodes.

The social engineering scam pages all look fairly similar, with some minor differences and varying levels of obfuscation. Some used regular HTML and JavaScript to render their scam page, while others obfuscated this content in order to make detection of this page non-trivial. Finally, when being prompted to download the Fake AV binary, some would host the content on the same host as the social engineering page while others would host the binary on a separate host.

The data set consists of 22,393 fetch logs. After filtering out fetch logs that did not yield a binary executable, we are left with 5,075 fetch logs. Analysis of this subset reveals patterns in the malware repositories. We were able to organize the repositories into groups by pattern matching on common file names, bulk subdomain service providers, and host strings. Using this repository grouping technique we identified six distinct MDNs. Two of them were very short lived and are discounted from the remainder of our analysis. To confirm that our grouping technique was accurate we looked for patterns in the resulting groups. The landing pages in each MDN contained injected redirection code that was distinct for each MDN. Further, the observed life spans of the repositories in each MDN had strong temporal correlations: only one repository was active at a given time for each MDN we identified. To illustrate the organizing effect of this procedure, we plot the repository and sample lifetimes for the entire dataset as a whole and then separately for each MDN. This is shown in Figure 3. Based on the fact that the MDNs were organized using only one of the three factors, and the resulting sets were also organized according to the remaining two factors, we are confident that our identification approach was accurate for the data in our study.

4.2 Behavioural differences between MDNs

For quick reference we have summarized the four Fake AV MDNs in Table 1 and refer to them by their MDN numbers.

Now that we've identified the various MDNs, we're better able to analyse each one in depth. One thing each MDN seems to have in common is the fact that each updates its landing pages so that they all point to the same destination at

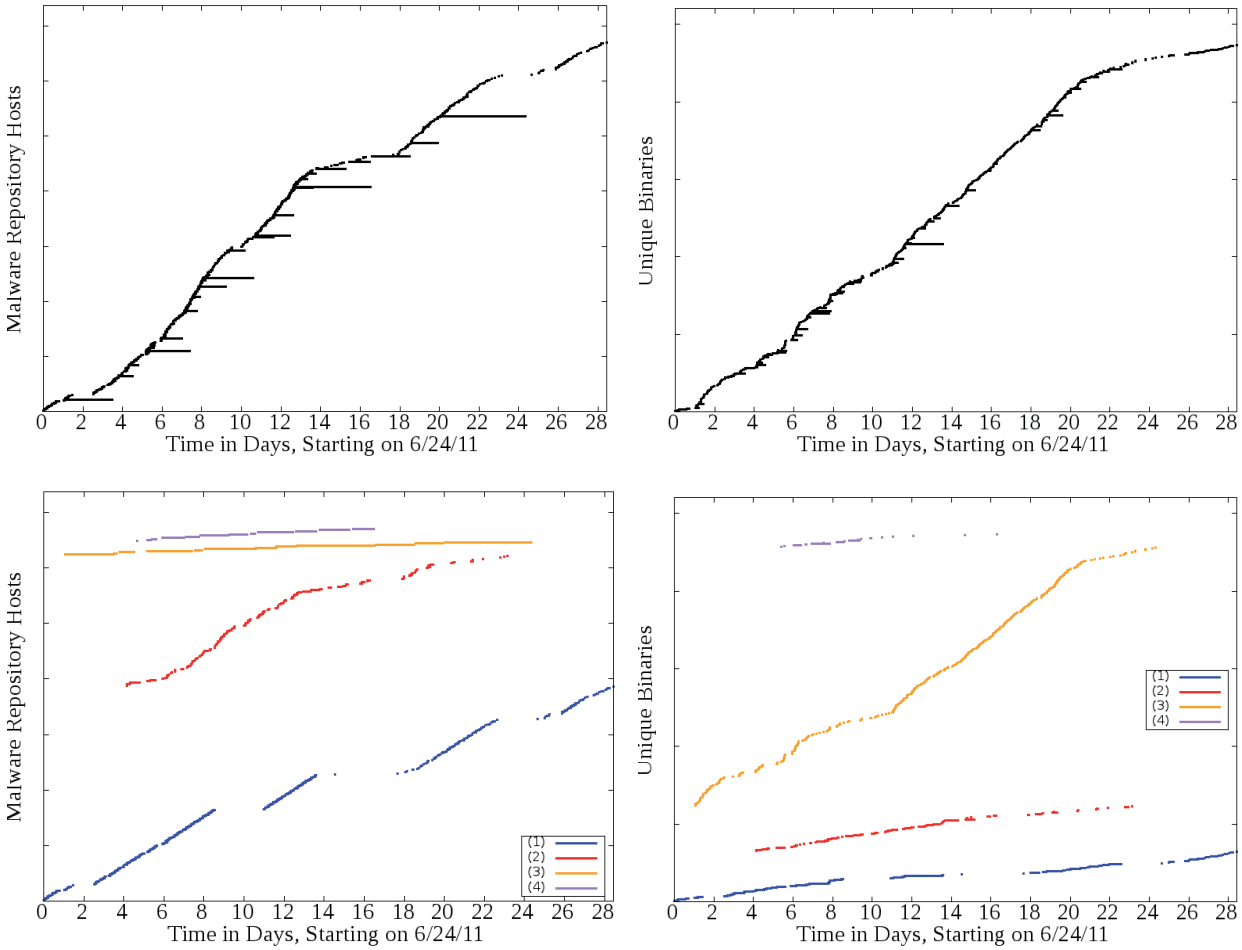

Figure 3: The graphs above show the lifetime of the malware repositories in the left column and the lifetime of the executable sample in the right column. The Y axis in all cases is discrete; each Y value represents a single repository/sample. The Y-axis for graphs in the top row are ordered by the first seen time of the repo/sample. The Y-axis for graphs in the bottom row are first ordered by MDN, and then ordered by first seen value, which reveals distinct patterns in the data set. It appears that each family has different repository and sample update patterns. This is confirmed when looking at this data in tabular form (see Table 2).

any given time. We know this because looking at the lifetimes of all our fetch logs, at no point did any of the lifetimes of malware repositories overlap with another belonging to the same MDN.

Of the four MDNs, MDN_1 was by far the most widespread as far as infected landing page counts were concerned. Compared to the next highest MDN, MDN_1 was almost nine times more prevalent based on our customer feedback results. Interestingly it also happens to be the longest running MDN, as the other three dropped from our results sets during the experiment.

Some characteristics of each MDN worth analysing include: the observed lifetime of the repositories; the observed lifetime of the binaries, which indicates the degree of polymorphism employed by the MDN; and whether a particular MDN appears to employ blacklisting (and if so what the observed blacklisting response is).

4.2.1 Repository update behaviours

We found two distinct patterns in the malware repository lifetimes. MDN_1 and MDN_2 took the approach of frequently changing the host of the malware repository, rotating them once every one to two hours, while the other groups updated far less frequently, every half day to two days. The main difference here is that when faced with a URL blacklist, the former groups are more resistant due to the frequency with which they update their hosts. It's worth pointing out that MDN_1 used a '.info' top-level domain (TLD) while MDN_2 used both the 'rr.nu' and 'findhere.org' bulk subdomain services. We found this odd as we expected an MDN with a low malware repository lifetime to use a cheaper source of malware repositories (e.g. through bulk subdomain services). MDN_3 opted for the cheapest route by simply using an IP address to host both the social engineering scam and Fake AV binaries, while MDN_4 also used a much less frequently updated list of '.info' TLD domains.

4.2.2 Binary update behaviours

We found that all MDNs periodically updated their binaries. MDN_3 appears to be using server-side polymorphism, as every request to their active repository results in a new binary executable. We manually verified this in case the lifetime was simply shorter than our re-fetch interval but greater than zero, and found that the binaries were in fact dynamically generated. With 333 binaries downloaded, this MDN

MDN #	Malware repo details	Binary update behaviour	Blacklisting?	Still active?
1	Randomly generated strings of .info TLD	Periodic updates	Confirmed, IP blacklisting, redirection to non-malicious sites	Yes
2	Initially hosted on findhere.org, then rr.nu	Periodic updates	None observed	No
3	Snowshoeing [18] through multiple ranges of IPs	Fully polymorphic	Possible	No
4	Static base string with incremented number appended, .info TLD	Periodic updates	None observed	No

Table 1: Summary of identified malware distribution networks ('still active' column is as of 24 July 2011).

MDN	Landing page count	Malware repository count	Repository lifetime average (s)	Repository lifetime σ (s)	Binary count	Binary lifetime average (s)	Binary lifetime σ (s)
1	347	193	4,875	2,794	64	19,937	24,886
2	39	118	3,783	6,235	59	10,064	12,152
3	19	12	156,493	113,675	333	0	0
4	8	21	69,766	49,816	17	10,879	21,649

Table 2: Malware distribution network statistics (σ = standard deviation).

produced the largest number of samples despite using the fewest malware repositories. The other three MDNs took a slightly different approach, with an average binary lifetime of between two and six hours and far fewer samples seen overall.

4.2.3 Blacklisting behaviours

MDN_1 and MDN_3 both showed signs of blacklisting, although only MDN_1 showed conclusive evidence of it. MDN_3 was an interesting case as it showed some indications of potential for blacklisting, though we never actually observed it in action. The following is a snippet of the redirection output from several infected landing pages in MDN_3:

```
document.write("<img src='//counter.yadro.ru/hit;Joh
nDeer?t52.6;r"+escape(document.referrer)+((typeof(sc
reen)=="undefined")?"":";s"+screen.width+"*"+screen.
height+"*"+(screen.colorDepth?screen.color-
Depth:screen.pixelDepth))+";u"+escape(document.
URL)+";"+Math.random()+"'"+"border='0' width='88'
height='31'>");
```

When this image is requested, it will send a fingerprint of the client back to yadro.ru which contains the referrer used, the screen resolution and pixel depth. It seems odd to profile the screen resolution. On further investigation we realized that a virtualized fetcher running in headless mode will produce a pixel depth of 0. This is a clear indication to anyone monitoring server logs that an automated crawler is harvesting the landing page. Another thing of note is that the path loading the image contained the string 'JohnDeer', which could be a reference to *John Deere*, a manufacturer of agricultural harvesting machinery ('harvest' being is a common industry term for collecting malware samples. Note that our experiments did not run in headless mode and therefore were not susceptible to this form of crawler fingerprinting.

MDN_1 exhibited blacklisting behaviour and was worth investigating further. We decided to run a separate experiment, using a new IP, that would fetch a landing page from MDN_1 as often as possible for two purposes: first, it provided more accurate measures of the repository and binary lifetimes, and, second, this aggressive re-evaluation interval is more likely to trigger a blacklisting response. We started the experiment on the afternoon of 30 June. The lifetime of malware repositories throughout the experiment remained fairly consistent; a consistent update pattern was visible until 2 July. At around 14:00 PDT the MDN began appending the query parameter '?q=av-sucks', to the normal server side 302 redirects. We speculate that, in addition to encouraging us, this query parameter was meant to fingerprint requests from our clients. Twelve hours later they had fully prevented us from accessing the malware repository; that is, the landing page would no longer redirect into the MDN. We tried changing a number of variables such as the referrer string, user agent, browser plug-ins installed, HTTP request headers, but none resulted in a successful fetch. At the same time, requests from different IP pools were successful, so we conclude that blacklisting was IP based.

In addition to the blacklisting incident described above, MDN_1 blacklisted our IP pools on several other occasions. Figure 4 illustrates the time line of these incidents; each incident can be seen as a gap.

The other two MDNs did not appear to do any sort of blacklisting, though there were times when they redirected us to non-Fake AV content including sites trying to push generic pills and pay-per-click link sites. At no point did they stop serving us content altogether, and often the content served to us would randomly rotate through Fake AV, pills and other pay-per-click sites.

4.3 Evaluation and proposed optimizations

The malware repository and malicious executable are clearly being updated frequently, and the distribution network is sophisticated in terms of the countermeasures in place to thwart the actions of security researchers. This is a worst

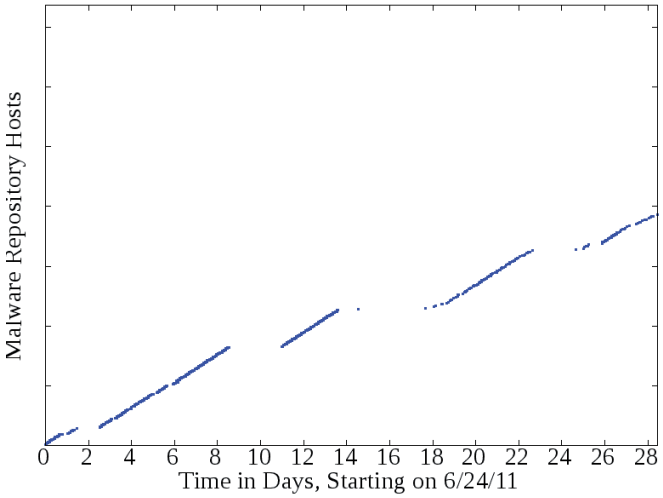

Figure 4: The malware repository lifetime of MDN_1. All gaps throughout the experiment occurred as a result of blacklisting. It wasn't until we rotated our proxy IP pool that we were able to continue fetching results.

case scenario where information needs to be frequently collected from the MDN in order to actively protect users, yet there is also a need to limit the exposure of our infrastructure. Based on the network behaviour presented above, we propose several techniques to reduce our client exposure to the network, which, we assume, will reduce blacklisting while maintaining a relatively high re-evaluation frequency.

The first technique leverages the high degree of fan-in from landing pages to malware repositories. Our data indicates that all landing pages in an MDN redirect to a single repository at a given time. For any MDN that satisfies this condition (i.e. one active repository at a time), a visit to a single landing page in the MDN will yield the active malware repository at a given time, and evaluations of the other landing pages are not required.

The second technique reduces the number of exposures to the repository, at the cost of higher uncertainty about the malicious executable being served by it. We propose the addition of a decision point during the evaluation of an MDN landing page. If the landing page redirects (directly or indirectly) to a known malware repository, and the repository has *recently* been visited, then the HTTP client should not make an HTTP request to the repository. This saves one exposure to the malware repository. The determination of *recent* is based on a threshold, which we refer to as the repository re-evaluation threshold (RRT). This second technique will be effective if at least one of two conditions is met:

1. The lifetime of an active repository in the MDN is long compared to the lifetime of a specific malware binary. Based on Table 3, this technique would be effective for MDN_1 and MDN_2, but not MDN_3 or MDN_4.
2. The ability of a researcher to proactively detect the malicious binaries (i.e. already detected without signature updates) is 'good enough'; revisiting the repository on every evaluation of the MDN is unnecessary. In this scenario, the MDN is being monitored to ensure the repositories are blocked and the samples do not begin breaking detection.

To evaluate these techniques, we simulate the re-evaluations that would be made with several combinations of the proposed techniques. The simulation is performed using the data we collected. To perform the simulation we assume that all landing pages in an MDN redirect to the same repository at a given time. As an example, assume an MDN has two landing pages, LP_1 and LP_2, and two repositories, R_1 and R_2, and that we have data points showing that at time 10: $LP_1 \rightarrow R_1$ and at time 20: $LP_2 \rightarrow R_2$. It is possible during simulation that an algorithm makes a fetch to LP_2 at time 10 and to LP_1 at time 16. The simulation will return $LP_2 \rightarrow R_1$ at time 10 (based on evidence from the LP_1 data point), and $LP_1 \rightarrow R_2$ at time 16 (based on the nearest data point: LP_2 at time 20).

For each simulation we calculate the number of fetches that would be performed, the number of exposures to landing pages and to repositories, the number of repositories discovered, and the number of executables discovered. Table 3 shows the simulation results of a fixed interval scheduler that does not implement the proposed techniques. For each simulation the re-evaluation interval is increased. We

Interval	Fetches	Landing page exposures	Repository exposures	Repositories discovered	Binaries discovered
Actual data	22,393	22,393	22,393	344	473
1 hour	276,161	276,161	276,161	317	384
2 hours	138,254	138,254	138,254	295	304
4 hours	69,310	69,310	69,310	194	226
8 hours	34,842	34,842	34,842	129	159
12 hours	23,119	23,119	23,119	99	126
1 day	11,762	11,762	11,762	64	79
2 days	6,074	6,074	6,074	41	45
7 days	1,999	1,999	1,999	16	16

Table 3: Impact of re-evaluation interval on MDN coverage. This set of simulation results illustrates the impact of increasing the re-evaluation interval on the repository and executable discovery rates. This shows that for Fake AV MDNs the network coverage drops significantly as the re-evaluation interval is increased. This is due to the highly dynamic nature of the MDN used to spread Fake AV.

Interval	RRT	Fetches	Landing page exposures	Repository exposures	Repositories discovered	Binaries discovered
Actual data	--	22,393	22,393	22,393	344	473
No optimizations, 1 hour interval	--	276,161	276,161	276,161	317	384
1 hour	0	2,905	2,905	2,905	322	379
1 hour	1 hour	2,905	2,905	2,905	322	379
1 hour	2 hour	2,905	2,905	2,905	322	377
1 hour	2 hours†	2,905	2,905	1,724	322	309
1 hour	3 hours	2,905	2,905	1,795	322	372
1 hour	4 hours	2,905	2,905	1,676	322	374
1 hour	8 hours	2,905	2,905	1,512	322	369
1 hour	12 hours	2,905	2,905	1,455	322	368
1 hour	24 hours	2,905	2,905	1,409	322	366

†*In all other simulations the RRT for MDN_3 was set to 0. For this simulation is was set to two hours. The reader will note the significant drop in sample discovery rate that results when optimization two is applied to an MDN that has polymorphic malware repositories. This is discussed further in the Discussion section.*

Table 4: Impact of optimizations on MDN coverage. The application of the first proposed optimization (row 3) results in a drastic (99%) reduction in the fetch volume required to profile the MDN (compared to row 2). The second optimization (rows 4–11) produces greater savings, this time in the number of exposures to the malware repositories; however, the second optimization does have a small cost in terms of missed samples. (RRT: repository re-evaluation threshold.)

see that the coverage (i.e. the number of repositories and binaries discovered) drops off quickly as the re-evaluation interval increases. This firmly establishes the need for frequent re-evaluation of the MDN in order to maintain coverage of the repositories and executables.

Table 4 shows the impact of the two optimization techniques. A naïve algorithm with a re-evaluation interval of one hour makes 276,161 requests during the simulation, versus only 2,905 requests made by a re-evaluation algorithm that uses knowledge of the MDN when re-evaluating landing pages. The cost of this optimization is a loss of five samples.

The application of the first optimization to our simulated re-fetching algorithm resulted in a fetch volume reduction of 99% with a 10%[2] loss in the sample discovery rate (excluding samples from the polymorphic MDN). When both optimizations are applied the number of exposures to the repositories drops by as much as 50% versus using just the first optimization; however, it is clear that the second optimization comes with a cost in terms of sample discovery. The impact of the reduced discovery rate on the vendor depends on the specifics of their products.

5. DISCUSSION

Based on the average lifetime of the executable binaries, we conclude that using checksum-based blocking as a stop gap for generic detection will only be effective for a subset of the MDNs we investigated. Tracking the average binary lifetime for each MDN in conjunction with the zero-day detection rates of the samples collected allows us to improve our detection triage process. When detection drops on samples from a purely polymorphic MDN, this requires immediate analyst attention. When detection dips on an MDN whose samples are longer lived, then an automated checksum approach is sufficient to reduce the urgency of the incident while analysts address the drop in detection rate.

Grouping samples by MDN helps analysts identify patterns to improve generic detections. For MDN_1, we detected 100% of the binaries through a single *Sophos* detection. However, the samples from the other MDNs were detected by no less than six distinct *Sophos* detections. After providing the grouped samples to analysts, they were able to quickly produce single generic detections per MDN.

Throughout our experiments, for each MDN all landing pages pointed to a single malware repository at any given time. From this we conclude that re-evaluating multiple landing pages belonging to the same MDN is a waste of resources, and worse, this activity exposes your fetching infrastructure to the MDN unnecessarily. Analysing the lifetime of the malware repositories and assigning a re-evaluation interval for each MDN will help minimize the resources required to monitor each threat, and reduce the chance of being blacklisted.

We observed that the technique of blacklisting is a real threat and is actively applied by the administrators of MDNs. This is an important phenomenon to consider when designing a system to monitor Fake AV. In the same way that security vendors monitor threats and blacklist large IP ranges, so too can malware distributors. Repeated visits to MDNs from IP addresses in the same range are easy to spot in server logs. We have seen the blacklisting response happen in under a

[2] To calculate this sample cost we first exclude the samples from the purely polymorphic MDN. 10% = (379 – 333) / (384 – 333).

week. We feel it is important for security vendors to use large pools of IP addresses spread across a variety of networks.

There are several avenues to solve the blacklisting problem. One idea is to use web appliances running on customer machines as a proxy (e.g. through an opt-in feature). This would provide security vendors access to the large pool of customers' IP address space for the purpose of malware sample collection. This way the blacklisting scenario is turned on its head: when an IP of a customer is blacklisted, it essentially is protected – in effect, the ideal outcome. Of course there are legal ramifications to be considered, as well as the risk of customers being targeted by MDNs in a retaliatory manner.

Another approach to the problem of blacklisting is increased collaboration among security vendors to share resources and eliminate duplicated data collection effort. URL and sample sharing among security vendors is already common practice; however, these arrangements do nothing to actually pool resources and reduce the overall number of exposures to the MDNs. It is unclear whether such collaboration will naturally emerge, especially since some vendors might view their ability to crawl from a large pool of IPs as their competitive advantage.

A final avenue worth pursuing is increased cooperation with organizations that have Internet-level views, such as ISPs or large research organizations. It has been shown that some MDNs pre-compute their repositories in advance. However, in other cases landing pages are periodically updated, via a pull or push mechanism, with the new repository. If these flows could be identified through passive network analysis, this would provide yet another means to reduce exposure to the MDN.

6. RELATED WORK

There have been many recent studies of malware on the web, of malware distribution networks, and specifically of the problem of fake anti-virus software. We briefly detail works that are most relevant to this study.

The seminal work by Provos et al. [6] has provided a foundation, both in terms of the methodology and the presentation of studies of malware on the web. Recent work by this group [19] specifically addressed Fake AV distribution networks. The results in this work were consistent with the observations we made in this paper. While their analysis typically presents results at the macro scale – something only possible with the visibility of an organization like *Google* – our work provides a very focused study of several MDNs and provides specific strategies for identifying and re-evaluating these MDNs. Recent work by Stone-Gross et al. [2] also focused on Fake AV networks, however their work focused primarily on the payment systems in place to monetize the infections, whereas we focus on the delivery networks.

Recent work by Zhang, Seifert et al. [20] studied a corpus of several billion fetch logs and showed ways to identify the MDNs. In this way their research is very similar to ours. Their approach incorporates more network information to identify MDNs and also provides a degree of automation to the process through the use of AutoRE [19]. Our work differs from theirs in that we use the identification of MDNs to adjust and optimize re-evaluation logic, whereas they used the identification of MDNs to retroactively identify malicious fetch logs to improve URL blocklists.

7. CONCLUSION

This paper presented a study of several malware distribution networks responsible for distributing Fake AV software by using social engineering tactics. We have pinpointed several solutions to identify distinct malware distribution networks (MDNs) and highlight the specific behaviours of the MDNs we have identified. Additionally we have shown that these behaviours can be leveraged to drastically reduce the amount of generated crawling traffic needed to track each MDN over time.

ACKNOWLEDGEMENTS

We would like to thank Adriana Iamnitchi for providing us *PlanetLab* access. We also want to thank all the analysts in *SophosLabs* for sharing their vast expertise.

REFERENCES

[1] Rajab, M.; Ballard, L.; Mavrommatis, P.; Provos, N.; Zhao, X.w. The Nocebo Effect on the Web: An Analysis of Fake Anti-Virus Distribution, Large-Scale Exploits and Emergent Threats. LEET 2010.

[2] Stone-Gross, B.; Abman, R.; Kemmerer, R.; Kruegel, C.; Steigerwald, D.; Vigna, G. The Underground Economy of Fake Antivirus Software. 10th Workshop on Economics of Information Security (WEIS), June 2011.

[3] Cluley, G. Phishing in a World of Warcraft. http://nakedsecurity.sophos.com/2011/01/20/phishing-in-a-world-of-warcraft/.

[4] Google Trends. http://www.google.com/trends/.

[5] Howard, F.; Komili, O. Poisoned search results: how hackers have automated search engine poisoning attacks to distribute malware. http://www.sophos.com/en-us/why-sophos/our-people/technical-papers/sophos-seo-insights.aspx.

[6] Provos, N.; Mavrommatis, P.; Rajab, M.; Monrose, F. All Your iFRAMEs Point To Us. USENIX Security Symposium 2008, pp.1–16.

[7] Komili, O. Fake AV, now for Windows 7! http://nakedsecurity.sophos.com/2010/03/04/fakeav-windows-7/.

[8] Wisniewski, C. Fake Firefox Warnings Lead to Scareware. http://nakedsecurity.sophos.com/2011/05/30/fake-firefox-warnings-lead-to-scareware/.

[9] Wisniewski, C. Mother's Day Search Terms Lead to Mac Rogue Security Software. http://nakedsecurity.sophos.com/2011/05/07/mothers-day-search-terms-lead-to-mac-rogue-security-software/.

[10] Google Code. http://code.google.com/more/.

[11] Star Trek Wiki – Memory Alpha. http://memory-alpha.org/wiki/Tachyon_detection_grid.

[12] Zeeuwen, K.; Ripeanu, M.; Beznosov, K. Improving Malicious URL Re-Evaluation Scheduling Through an Empirical Study of Malware Download Centers. Joint WICOW/AIRWeb Workshop on Web Quality (WebQuality 2011), 2011.

[13] Zeeuwen, K. 2011. Optimizing Re-Evaluation of Malware Distribution Networks. Masters Thesis, University of British Columbia, Vancouver, Canada.

[14] RefControl. http://www.stardrifter.org/refcontrol/.

[15] Project Sikul. http://www.sikuli.org/.

[16] PlanetLab. http://www.planet-lab.org/.

[17] jsunpack. http://jsunpack.jeek.org/dec/go/.

[18] Snowshoe spamming. Spamhaus. http://www.spamhaus.org/faq/answers.lasso?section=Glossary#233.

[19] Xie, Y.; Yu, F.; Achan, K.; Panigraphy, R.; Hulten, G.; Osipkov, I. Spamming botnets: Signatures and characteristics. SIGCOMM 2008.

[20] Zhang, J.; Seifert, C.; Stokes, J.; Lee, W. ARROW: Generating Signatures to Detect Drive-By Downloads. WWW, 2011.

[21] Howard, F. Malware with your Mocha? Obfuscation and anti-emulation tricks in malicious JavaScript. http://www.sophos.com/en-us/why-sophos/our-people/technical-papers/malware-with-your-mocha.aspx.