Manufacturing Compromise: The Emergence of Exploit-as-a-Service

Chris Grier[†◇] Lucas Ballard[□] Juan Caballero[§] Neha Chachra[*] Christian J. Dietrich[II]
Kirill Levchenko[*] Panayiotis Mavrommatis[□] Damon McCoy[‡] Antonio Nappa[§]
Andreas Pitsillidis[*] Niels Provos[□] M. Zubair Rafique[§] Moheeb Abu Rajab[□]
Christian Rossow[II] Kurt Thomas[†] Vern Paxson[†◇] Stefan Savage[*] Geoffrey M. Voelker[*]

† University of California, Berkeley * University of California, San Diego □ Google
◇ International Computer Science Institute § IMDEA Software Institute
II University of Applied Sciences Gelsenkirchen ‡ George Mason University

ABSTRACT

We investigate the emergence of the *exploit-as-a-service* model for driveby browser compromise. In this regime, attackers pay for an exploit kit or service to do the "dirty work" of exploiting a victim's browser, decoupling the complexities of browser and plugin vulnerabilities from the challenges of generating traffic to a website under the attacker's control. Upon a successful exploit, these kits load and execute a binary provided by the attacker, effectively transferring control of a victim's machine to the attacker.

In order to understand the impact of the exploit-as-a-service paradigm on the malware ecosystem, we perform a detailed analysis of the prevalence of exploit kits, the families of malware installed upon a successful exploit, and the volume of traffic that malicious web sites receive. To carry out this study, we analyze 77,000 malicious URLs received from Google Safe Browsing, along with a crowd-sourced feed of blacklisted URLs known to direct to exploit kits. These URLs led to over 10,000 distinct binaries, which we ran in a contained environment.

Our results show that many of the most prominent families of malware now propagate through driveby downloads—32 families in all. Their activities are supported by a handful of exploit kits, with Blackhole accounting for 29% of all malicious URLs in our data, followed in popularity by Incognito. We use DNS traffic from real networks to provide a unique perspective on the popularity of malware families based on the frequency that their binaries are installed by drivebys, as well as the lifetime and popularity of domains funneling users to exploits.

Categories and Subject Descriptors

K.4.1 [**Public Policy Issues**]: ABUSE AND CRIME INVOLVING COMPUTERS

Keywords

Security, Malware

CCS'12, October 16–18, 2012, Raleigh, North Carolina, USA.

1. INTRODUCTION

In this work we investigate the emergence of a new paradigm: the *exploit-as-a-service* economy that surrounds browser compromise. This model follows in the footsteps of a dramatic evolution in the world of for-profit malware over the last five years, where host *compromise* is now decoupled from host *monetization*. Specifically, the means by which a host initially falls under an attacker's control are now independent of the means by which an(other) attacker abuses the host in order to realize a profit. This shift in behavior is exemplified by the *pay-per-install* model of malware distribution, where miscreants pay for compromised hosts via the underground economy [4, 41]. Where the pay-per-install market relies on a mixture of social engineering, spam, and other infection vectors to compromise hosts, the exploit-as-a-service model specifically relies on *driveby downloads*.

Our prior work suggests that driveby downloads that target browser and plugin vulnerabilities (e.g., PDF viewers, Flash, and Java) to install malware now represent the largest threat to end users [30]. The vanguard of this assault is lead by the development of *exploit kits*: packages of browser exploits that simplify the act of compromising victims that visit malicious websites. While web exploit kits themselves are not new, dating back to at least MPack in 2006 [34], there is little doubt that exploit kits have come of age. The recent compromise of `mysql.com`—a site in the Alexa 1000—was used to infect visitors using the Blackhole exploit service [14], which we have found anecdotally (via Blackhole management screenshots) to achieve a successful compromise rate of 9–14% [17, 43, 46]

In order to understand the impact of the *exploit-as-a-service* marketplace on the malware ecosystem, we perform a detailed analysis of the prevalence of exploit kits, the families of malware installed upon a successful exploit, and the volume of traffic malicious websites receive. To carry out this study, we aggregate and analyze 77,000 malicious URLs received from Google Safe Browsing, and from a crowd-sourced feed of blacklisted URLs known to direct to exploit kits. For each of these URLs, we also obtain at regular intervals a copy of the malicious binaries they attempt to install, totaling over 10,000 variants from the course of March 1, 2012 until April 20, 2012.

We run each of these binaries in a contained execution environment and determine a sample's family as well as its *monetization approach*, such as spam, fake anti-virus, and a multitude of other strategies for profiting off of an infection. To offer a comparison to

other competing malware distribution techniques, we develop and acquire malware feeds that include malicious email attachments, torrents for pirated software, malicious binaries installed by droppers tied to the *pay-per-install* marketplace, and binaries extracted from live network traffic. We find that drivebys and droppers are the primary source of the most prominent malware families, indicating a continuing shift in the malware ecosystem towards miscreants that specialize solely in compromising hosts.

In addition to the malware installed by browser exploits, we examine the exploit kits that are behind the scene. We determine that Blackhole accounts for *29% of all malicious URLs*, followed in popularity by Incognito and a small handful of other exploit kits. Combined, these kits are used to distribute at least 32 different families of malware. Furthermore, we map out the complex infection chain tied to driveby exploits, including the use of compromised pages and the redirection of victims to multiple exploit kits simultaneously.

Finally, using 3.5TB of *passive DNS* data collected from several large ISPs and enterprises, we provide a unique perspective on the ranking of malware families based on the frequency that drivebys install their binaries as well as the lifetime of exploit domains. We find that droppers, information stealers, and fake anti-virus software dominate the monetization of drivebys. Despite finding that exploit domains survive for a median of *only 2.5 hours*, we show that thousands of visitors suffer exposure to drivebys due to the compromise of popular webpages. Lastly, we examine the impact of Google Safe Browsing on driveby domains. While our analysis clearly highlights that websites hosting driveby exploits encounter immense pressure, this does not suffice to disrupt operations completely.

In summary, we frame our contributions as follows:

- For each driveby site, we identify the most popular exploit kit used and the malware family served by the site, including its monetization scheme.
- Using passive DNS data we estimate the *relative popularity* of malware families distributed via driveby exploits.
- We report on operational aspects of driveby compromises; specifically, the lifetime of domains used to distribute malware (hours on average), and the effectiveness of interventions like Google Safe Browsing.

We organize the rest of this paper as follows. Section 2 provides a background on exploit services and host compromise. Section 3 describes our measurement and malware classification methodology. In Section 4 we present our results. We frame related work in Section 5 and summarize the paper in Section 6.

2. BACKGROUND

From the time a victim accesses a malicious website up to the installation of malware on their system, there is a complex chain of events that underpins a successful driveby download. We present a sample infection chain for a real driveby exploit in Figure 1, obfuscating only the compromised website that launched the attack. The process begins with a victim visiting a compromised website or otherwise malicious page (❶). The victim's browser receives a series of redirections through a chain of intermediate pages (❷) that obscures the final landing page, which hosts an exploit kit (❸). This final page attempts to exploit a victim's browser, targeting either vulnerabilities in the browser, or browser plugins such as Adobe Flash and Java. If an unpatched version of the vulnerable software exists, the victim's machine is compromised and any variety of malware can be installed (❹,❺).

The challenge of identifying new browser exploits, funneling traffic to malicious webpages, and monetizing compromised hosts has led to a diversification of roles within the malware ecosystem, and in particular, the emergence of a new marketplace surrounding exploit-as-a-service. This marketplace currently includes two business models: *exploit kits* and *Traffic-PPI* services.

In the exploit kit model, miscreants either purchase *exploit kits* (software only) or rent access to pre-configured *exploit servers* (hardware and exploit software). This business model fulfills all the requirements of step ❸ and ❺ in the driveby chain. Clients are responsible for luring their own victims and determining which malware to distribute.

Traffic-PPI service take the exploit pack model one step further and can be considered an evolution of the *pay-per-install* service model [4]. In this model, clients simply *purchase installs* and provide their binaries (❹), while the Traffic-PPI service takes care of the entire process of generating traffic, redirecting, and exploiting a victim's browsers (❶, ❷, ❸) until finally installing the client's software (❺).

2.1 Exploit Kits and Servers

For our purposes we will use the term *exploit kits* (or *packs*) to refer to software packages that bundle multiple exploits targeted at vulnerabilities in web browsers and their plugins (e.g., Flash, PDF and Java). Popular exploit kits include Blackhole, Eleonore, and Phoenix [9]. Attackers install exploit kits on web servers, and we term the combination of server plus exploit kit as an *exploit server*. Upon a visit to a domain hosted in an exploit server, the exploit kit automatically profiles the browser and delivers an exploit based on the operating system, browser, and plugin configuration. If the exploit succeeds, it downloads a binary that then executes on the user's computer.

Exploit kits date back at least to MPack from 2006 [34]. The traditional business model for commercializing exploit kits has been one-time fees [27]. Like traditional software, once purchased, such licenses generally allow unlimited use, although some kits attempt to enforce limits based on time or domains and IP addresses used by the exploit server. For example, the Phoenix exploit kit charges a flat fee for a single domain license. This license cost $400 in November 2009 and current versions start at $2,200 [23].

Recently, the Blackhole exploit kit introduced the exploit server model, which can be viewed as an application of the software-as-a-service paradigm for browser compromise. With an exploit server, an exploit kit author provides clients with access to a pre-imaged host running their exploit software, which clients can configure to drop various malware variants; all other aspects of hosting, including domain registration and periodic exploit updates, are handled by the exploit author. Renting a Blackhole server ranges from $50 for an hour up to $500 for a month (a surprisingly narrow range in price for which we do not have an explanation), which includes full support for the duration of the rental [13].

2.2 Traffic-PPI Services

Traffic-PPI services are a natural evolution of the classic pay-per-install (PPI) service [4] adapted to the drive-by-download ecosystem. The key difference between the models is that in the classic PPI model, the PPI service outsources the task of exploiting the target hosts to affiliates. Affiliates rely on their own distribution methods, which include torrents, drivebys, spam, or, surprisingly, simply using other PPI services. In contrast, the service that runs a Traffic-PPI marketplace is in charge of exploitation exclusively

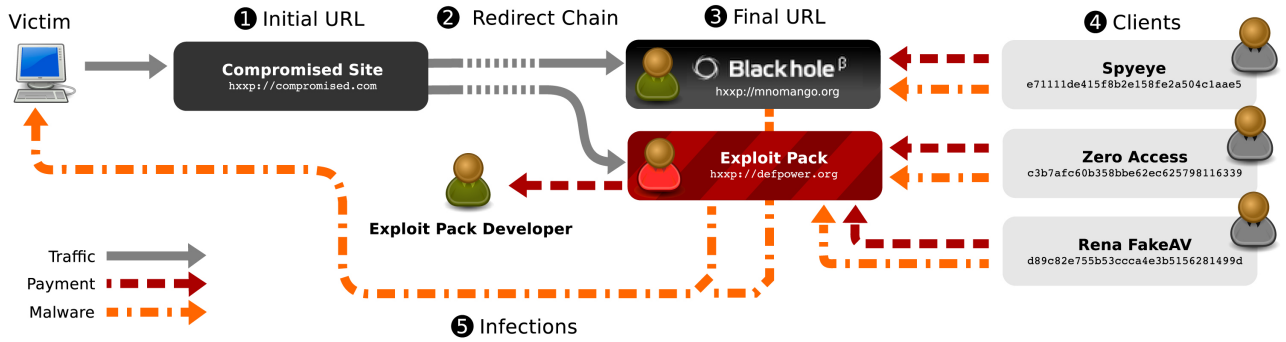


Figure 1: The drive-by-download infection chain. Within the exploit-as-a-service ecosystem, two roles have appeared: *exploit kits* that aid miscreants in compromising browsers (❸), and *Traffic-PPI* markets that sell installs to clients (❹) while managing all aspects of a successful exploit (❶,❷,❸).

through drivebys. Often, these Traffic-PPI services rely on affiliates to provide traffic. Similar to web advertising, these services charge clients a cost-per-impression (CPI) for installing a client's program and pay affiliates for traffic that leads to a successful infection. Traffic-PPI services can absolve themselves of further work by outsourcing the creation of browser exploits to an exploit pack author, where the Traffic-PPI simply becomes a middle man between traffic generators, exploit kit writers, and clients. We present two examples of Traffic-PPI services to illustrate their business models.

SellMeYourTraffic: The Traffic-PPI program SellMeYourTraffic states that they buy any type of traffic from affiliates, regardless of quality, paying affiliates between $0.80–$3.00 USD per thousand visits [37]. In their own words: "Send it all... Pop-ups, pop-unders, blind clicks, re-directs, arbitrage, remnant traffic, and whatever else you have [. . .]" Similar to other Traffic-PPI affiliate programs, they do not specify how they will use the traffic, but behind the scenes they redirect incoming traffic to instances of the Blackhole exploit kit [38]. For instance, when a user visits `feekiller.com`, a website belonging to a SellMeYourTraffic affiliate, the user ends up being redirected to the SellMeYourTraffic service at `click.icetraffic.com` and eventually to a Blackhole exploit server. This exchange was captured by URLQuery.net, a service that visits websites and provides detailed logs for each visit [38].

Traffbiz: Traffbiz is a Russian Traffic-PPI service. They also run an affiliate service and pay affiliate webmasters $1.6 USD per thousand visits from users in Russia [45]. To perform the hand-off of traffic, the affiliate receives a small snippet of JavaScript code to add to their websites. The JavaScript snippet contains a link to a banner picture, which in turn includes a five digit unique affiliate identifier. In reality, the banner is the start of a redirect sequence that eventually leads the user's browser to an exploit server running the Blackhole exploit kit. Users outside of Russia are simply presented with a normal banner.

2.3 Monetization Approaches

Profit lies at the heart of the modern malware ecosystem. Once a machine has been successfully infected, miscreants have a number of monetization vectors at their disposal, many of which in fact reflect services that further fuel the underground economy. We provide a brief overview of these techniques, paying particular attention to the most notorious families of malware.

Spamming: Spam pertains to any form of bulk unsolicited messaging, including messages sent via email and social networks. Examples of well-understood email spam botnets include Rustock [16], Storm [19], and MegaD [6], while a number of unidentified programs target Facebook and Twitter [11,44]. Monetization comes in many forms, including the sale of pharmaceuticals, replica goods, and fake software [21]. While a diversity of roles exists within the spam marketplace, in the context of monetization approaches when we refer to spam we specifically mean the bulk distribution of messages.

Information Theft: Information stealers such as Torpig [40], Zeus, and SpyEye harvest sensitive user data from compromised machines, including documents, passwords, and banking credentials. These in turn can be cashed out or sold to the underground economy of *carders* [10]. The existence of crimeware toolkits [18] readily available for purchase has led to a proliferation of customized botnets, in contrast to the monolithic botnets associated with spam.

Clickfraud: Clickfraud generates revenue by using automated bots to simulate legitimate traffic to pay-per-click advertisements [25, 42]. These ads typically appear on pages controlled by miscreants, while the ads are syndicated from advertising networks such as Google AdSense.

Browser Hijacking: Browser hijacking includes any malware that overwrites the default browser search engine or re-routes web traffic on a machine in order to garner traffic to a particular website. Third parties can purchase this traffic either seeking to bootstrap infections (e.g., by directing traffic to driveby downloads) or for other revenue streams, such as advertising or product scams. We exclude malware that hijacks browsing sessions to steal credentials or display phishing pages from this category, as the goal of such variants is to siphon data for information theft, not to generate traffic.

Fake Software: Fake software includes any malware that prompts a user to install or upgrade ineffectual software. The most prominent approach here is selling rogue antivirus [31, 39]. These programs prompt users to pay a one-time fee in order to remove nonexistent malware infections, providing no protection in return.

Proxies & Hosting: For externally facing hosts, miscreants can convert compromised machines into bulletproof hosting services and proxy networks that are re-sold as tools for other miscreants [12]. These machines can in turn host vital C&C infrastructure, act as anonymizers, or simply provide a diverse pool of IP addresses to circumvent IP-based restrictions for sending spam or registering accounts.

Droppers: Rather than employing one of the aforementioned techniques, miscreants can sell successful infections to other parties on the underground market [4]. In some cases, this involves miscreants installing a *dropper* on a compromised host. This tool automatically

Feed	*Vector*	*Start*	*End*	*Num. Run*
Google	Driveby	4/2012	5/2012	4,967
BH/Phoenix	Driveby	3/2012	5/2012	5,341
Sandnet	Dropper	9/2011	5/2012	2,619
Spam Traps	Attachment	2/2012	5/2012	2,817
Torrents	Warez	9/2011	5/2012	17,182
Arbor ASERT	Live Traffic	8/2011	5/2012	28,300

Table 1: Malware feeds used in the study.

contacts a *pay-per-install* provider for new binaries to install in exchange for a fee. We differentiate between *staged installers* where an initial infection is used to bootstrap updates or new features, and droppers, where control of a machine is re-sold to a second or even multiple parties.

3. METHODOLOGY

Our study of driveby downloads centers around the malware installed upon a successful browser exploit. Figure 2 shows our pipeline for executing malware to determine its family and monetization. We receive malicious binaries from a variety of feeds. While in this paper we focus on malware distributed through driveby downloads, we also analyze competing infection vectors to explore the prominence of driveby downloads.

We execute a sample of these binaries in a contained environment, tracking each sample's behavior. We then automatically cluster the behavioral reports of variants and provide them to analysts for manual classification.

3.1 Malware Feeds

Our malware collection infrastructure consists of six distinct feeds. Five of them capture traffic from specific infection vectors. These feeds include malicious executables installed after a successful driveby download exploit (two distinct sources); executables obtained by infiltrating the pay-per-install marketplace; malicious email attachments; and infected executables embedded in cracked or pirated software downloaded from torrents. The final feed constitutes a collection of binaries found in the wild by honeypots and network traffic monitors, as explained below.

Obtaining a representative sample of malware presents an arduous, if not insurmountable, task [5]. Instead, we aim to capture binaries that span the breadth of popular infection vectors, allowing us to characterize differences in monetization approaches across each vector. Table 1 summarizes of our six feeds and the volume of binaries we acquire from each. We now discuss each in turn.

Driveby Downloads: Our first feed of driveby downloads consists of executables and their corresponding URLs found by Google's Safe Browsing infrastructure [29]. Google uses a web browser to visit suspicious web sites, flagging URLs that attack the browser and result in malicious side effects. We receive a copy of the malware that the attack installs, along with the URL responsible for the exploit and URL for the downloaded malware. In total, we received 4,967 binaries this way, all of which we presume as malicious due to the nature of their installation.

Our second driveby download feed focuses on payloads delivered by instances of the Blackhole and Phoenix exploit packs. We rely on crowd-sourced identification of driveby URLs [24], which we then visit at regular intervals to download the binary that it currently serves to infection victims. Using this approach, we acquired 5,341 additional driveby-related binaries.

Droppers: We receive a feed of binaries from Sandnet [36], which automatically contacts dropper services [35] and requests the latest binaries that third-party miscreants paid to have installed. In total, we received 2,619 binaries, which we use as a sample of the activity occurring in the pay-per-install market.

Email Attachments: Email attachments have a long history as a vector of malicious software. We receive a feed of emails, sent to spam traps hosted on a variety of domains, from which we extracted 2,817 attachments corresponding to either archive files (ZIP, RAR) that contain a Windows executable or plain Windows executables.

Torrents: Torrents used to distribute pirated software, cracks, and patches represent another common method for infecting end users. To analyze this vector, we scan ThePirateBay, isohunt, h33t, and a number of other trackers every four hours, downloading any new software torrents. (When we download content, we prohibit uploading to avoid enabling further infringement.) In total, we downloaded 17,182 torrents that constitute either archive files or Windows executables. As not all of these are necessarily malicious, we rely on clustering and manual classification, as discussed below, to identify harmful samples.

Live Network Traffic: Our final feed consists of 28,300 executables identified by the Arbor Security Engineering and Response Team and provided via their *ATLAS* system.[1] The feed does not target any one specific infection vector, but rather encompasses executables found throughout the wild, including URLs, honeypots, and in network traffic. Executables in the feed are not guaranteed to be malicious, again requiring a manual effort to filter out benign samples. We compare against this feed to determine attacks broadly seen in the wild versus those found specifically in drivebys.

Feed Overlap: A comparison of the MD5s of samples in each feed reveals that 0.4% of the binaries appear in more than one of the feeds. The most frequently occurring overlap (0.3%) are samples in the Sandnet and the Arbor ASERT feeds. Comparing feeds by MD5 sum is problematic, as frequent (and even on-demand) repacking of malware causes numerous distinct samples of the same malware to occur in the wild [4]. In Section 4 we provide another comparison of feeds based on the classification of the samples.

3.2 Contained Execution

We execute each binary in a virtualized environment provided by the GQ honeyfarm [20], which supports monitoring malware execution while providing a flexible network policy. We use Windows XP Service Pack 3 for all executions, and the system can process thousands of binaries per day.

In our experiments the execution environment prevents all traffic from directly contacting outside hosts, and instead redirects traffic to internal services. To provide realistic services that mimic the destinations that malware samples contact, we have built several internal services that respond to DNS, HTTP, and SMTP requests on demand. Our HTTP server operates as a transparent HTTP proxy for hosts on a whitelist, and replies with HTTP 200 OK messages and empty bodies for all non-whitelisted destinations. We handcraft the whitelist to include websites commonly used by malware as connection tests, where failed connections would otherwise result in the malware terminating prematurely.

For DNS, our service answers all queries, even requests without a valid answer, to ensure that domain takedowns have limited impact on malware execution. Finally, for SMTP traffic our server contacts the original destination and relays the server banners to the malware, but does not relay SMTP commands to the real server. For all other protocol types we provide a sink that will accept packets but does not respond.

[1] `http://atlas.arbor.net`

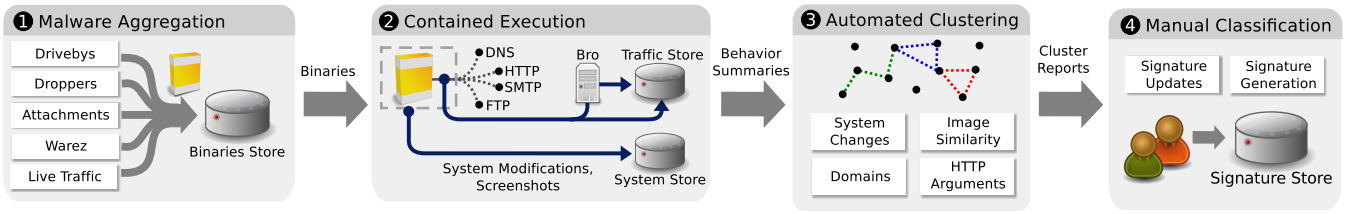


Figure 2: Architecture of our malware execution environment. We receive a feed of malware executables from a variety of sources (❶), which we execute in a contained, controlled environment that captures network traffic, system modifications, and screenshots before terminating (❷). We cluster the behavioral reports using a variety of techniques (❸) before finally supplying the clusters to analysts for manual classification (❹).

Feed	*DNS*	*HTTP*	*System*	*Windows*
Driveby	40.1%	44.7%	60.0%	45.1%
Dropper	28.6%	22.2%	56.8%	20.5%
Attachment	51.3%	5.1%	100.0%	31.6%
Torrent	4.1%	3.7%	64.5%	66.3%
Live	13.1%	9.9%	42.1%	39.1%

Table 2: Types of output produced by each feed. We use each output as a behavioral summary for clustering, if present.

In addition to network monitoring, our system collects operating system events, including process creation, file modifications, and registry changes. Prior to shutdown, we take a screen capture of the running system for use during the automated clustering step which leverages visible content.

3.3 Automated Clustering

In order to minimize polymorphism and guide analysts towards the most prominent families of malware, we cluster executions with similar behavioral signatures, as in our previous PPI work [4]. Our approach targets four characteristics: the domains a malware sample contacts, the HTTP requests it makes, any system modifications, and finally the layout of the windows it spawns, captured by a screenshot. Table 2 summarizes which classes of binaries perform each of the aforementioned actions and highlights why no single approach is sufficient.

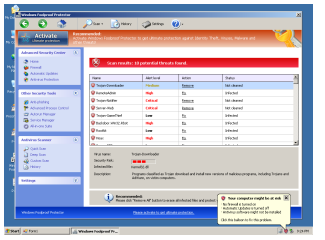
Our clustering draws from a long lineage of behavioral-based malware classification [2, 3, 33]. Building on this work, we make a number of improvements to clustering network traffic, as well as offer new techniques for clustering binaries with no system or network side-effects.

Domains: Domain-based clustering identifies binaries that contact the same hosts, whether they are C&C servers or test connections (such as a request to `google.com`). We tokenize domains by extracting increasingly specific subdomains. For instance, we tokenize `mail.lebanon-online.com` as {`com`, `lebanon-online.com`, `mail.lebanon-online.com`}. Once tokenized, we consider each token as an independent feature vector. We produce clusters by grouping binaries with identical feature vectors (e.g., all binaries that contact a `com` top-level domain, or that contact the `lebanon-online.com` domain). As a result, we can place a single binary into multiple clusters. To avoid some overly broad clusters, we omit clusters that match a manually generated whitelist of domains and tokens that are too generic, such as domains contacted by Windows during initialization, independent of the presence of malware. While we could consider more sophisticated approaches for clustering domains [48], our approach works well in practice, and analysts find it easy to interpret. However, due to the ephemeral nature of malicious domains, new clusters appear over time, which requires re-labeling. To avoid this limitation, we consider a number of other clustering approaches.

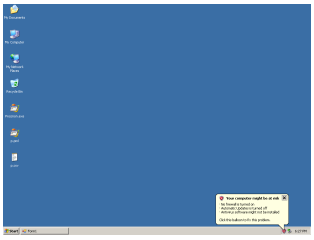
HTTP Arguments: For binaries that produce an HTTP request, we examine the User-Agent string tied to the request as well as GET and POST arguments. We produce tokens by splitting a request's parameters on any non-alphanumeric character. For instance, `/forum/showthread.php?page=5fa58` is tokenized as {`forum`, `showthread`, `php`, `page`, `5fa58`}. We then take the resulting set of tokens and generate all possible combinations, treating each unique combination as an independent feature vector. A cluster consists of all binaries with identical feature vectors (e.g., all binaries with HTTP requests containing just the tokens `showthread` and `5fa58`, with separate clusters for binaries containing all the aforementioned tokens). Our aim with this approach is to capture increasingly specific HTTP parameters, while at the same time accounting for natural variations to HTTP requests such as timestamps that would otherwise prohibit direct matching. The advantage of clustering on HTTP arguments is that while domains change due to takedown, the protocols for communicating with a C&C server are more stable and thus easier to cluster on.

System Modifications: We can also assess variation amongst malware families by the changes made to a system upon installation, including the names of spawned processes, the paths and values of new, deleted, or modified registry keys, and the paths of new, deleted, or modified files. Due to the adversarial nature of anti-virus signatures, many of these modifications will incorporate purposeful randomization to thwart matching. Nevertheless, we tokenize each of the aforementioned values based on non-alphanumeric characters and compute all possible combinations using the same approach as HTTP arguments. In order to avoid clustering on changes to the Windows system that result from benign Windows code that runs simultaneously alongside malware, we generate a whitelist of actions that occur across all binaries, including a manually labeled set of benign samples. We take care to not simply whitelist processes in the event that malware injects into a previously benign process.

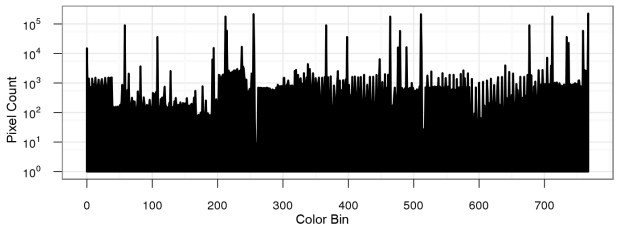
Screenshots: Our final approach to clustering groups binaries by a screen capture of the Windows machine after a binary sample has run its course, just prior to tearing down the virtual environment. Screen captures aid immensely with identifying fake anti-virus and other software that solicits a user response but does not produce any network activity or system changes in the absence of user interaction. Anderson et al.used image shingling combined with clustering to identify similar spam web pages [1]; similarly, we seek to cluster malware with similar UI components. For two screenshots m and n, we begin by computing each image's histogram, $H(m)$ and $H(n)$. A histogram, $H(m)$, contains the number of pixels, $x_{m,i}$,



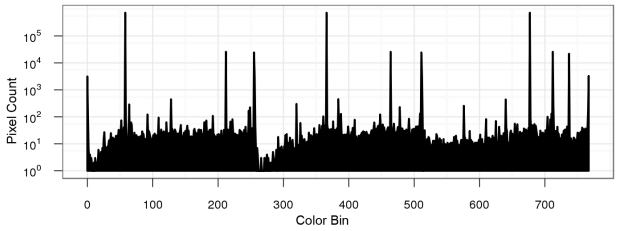
(a) Fake Anti-Virus Warning



(b) Blank Screen



(c) Fake Anti-Virus Histogram



(d) Blank Screen Histogram

Figure 3: Sample output of image clustering. We identify a number of fake software campaigns that otherwise produce no network activity or system modifications based on differences in screenshot histograms.

in each bin, i, corresponding to an 8-bit bin for each of the red, green, and blue color bands (each band is analyzed separately, totaling 768 bins). To compare the two images, we calculate the root mean squared deviation between the two histograms:

$$\text{RMSD}(H(m), H(n)) = \sqrt{\frac{\sum_{i=1}^{n}(x_{m,i} - x_{n,i})^2)}{n}} \quad (1)$$

If this difference is smaller than a manually determined threshold τ, we consider the two images as identical, and cluster them. We do not require exact duplication to allow subtle window variations such as different banners and text. Via feedback from analysts, we adopt a τ value of 100 for screenshots with a size of 1024x768. Figure 3 shows two clusters identified by this approach: a fake antivirus warning, and an executable with no windows, along with their histograms.

Discussion: Each of the clustering techniques was included because of the ability to discern specific features in malware. For example, the Fake WinZip family of malware (see Table 3) does not produce network traffic or system modifications; however, screenshot clustering quickly and accurately identifies this family of malware. System modifications were critical to identify Bitcoin mining bots, which have no UI components and produce only test network connections. We can easily identify the mining bots by the creation of a new file named "nvcuda.dll" (the NVIDIA CUDA library). For the majority of the families of malware we find, network features provide the most utility for clustering binaries.

3.4 Manual Classification

For the final step in our analysis pipeline, we perform manual analysis of cluster reports. Here, analysts (e.g., conscripted security researchers) review large clusters of binaries with similar behavior and compare them against public reports. Manual analysis is required to wade through the multitude of family names and the inaccuracy inherent in family labels. We also examine a sample's screenshot, file modifications, and network behavior to determine how a sample monetizes an infected host. In the event we could not locate a corresponding public family name, we apply a generic cluster label. Note that even in these cases, we can still identify the monetization technique for the cluster.

4. DRIVEBY DOWNLOAD ANALYSIS

In this section we provide a detailed investigation of the lifecycle of driveby downloads and the popularity of exploit kits. We find popular exploit kits including Blackhole and Incognito account for over 47% of all pages serving exploits. These kits are used to install at least 32 identifiable families of malware, which cover the spectrum of monetization techniques, highlighting the importance of the exploit-as-a-service market to the malware ecosystem. We compare the other infection vectors in our dataset to drivebys and find only droppers and live traffic carry similar malware variants.

Using passive DNS data, we examine the volume of traffic that domains hosting exploits receive, as well as their lifetime, finding exploit domains have a median lifetime of a *mere 2.5* hours before disappearing. We also assess the impact of Google Safe Browsing on the lifetime of drivebys, and conclude with trends we identify in the malware monetization techniques.

4.1 Comparison of Infection Vectors

We begin with a comparison of the binaries installed by driveby exploits and those found in other infection vectors. Table 3 shows a ranking of malware families based on the percent of binaries in each feed that belonged to a particular family. For instance, 12% of binaries installed by drivebys were Emit, a dropper that installs other malware families. We note that torrents and live traffic contain a great deal of goodware, resulting in low percentages of malware.

In terms of variants, driveby downloads are dominated by droppers, information stealers, and fake antivirus. Some of the same families such as Emit and ZeroAccess (otherwise called Sirefef) are also distributed by droppers, indicating that malware authors often choose multiple vectors to obtain infections. By contrast, our attachment feed is dominated by worms that are relics from nearly 9 years ago. While these families represent a sizeable number of binaries, their command and control infrastructure has long been dismantled, relegating the worms to merely propagating. Surprisingly, the vast majority of pirated software is benign. Those torrents that do contain malware are primarily focused on spyware and adware, with the exception of ZeroAccess.

We find that live traffic from Arbor represents a cross-section of all our feeds, though it only captures a subset of the prominent families. In summary, the most notorious malware variants currently active are found only in drivebys and droppers. Furthermore, no single feed is a panacea for obtaining a representative sample of all malware variants. This leaves a significant hurdle for malware research, requiring malware studies to infiltrate pay-per-install services, develop tools to identify driveby downloads, and simultaneously capture the plethora of other infection vectors used in the wild.

Rank	*Driveby*	*Dropper*	*Attachment*	*Torrent*	*Live*
1	Emit (12%)	Clickpotato (6%)	Lovegate (44%)	Unknown Adware.A (0.1%)	TDSS (2%)
2	Fake WinZip (8%)	Palevo (3%)	Mydoom (6%)	Sefnit (0.07%)	Clickpotato (1%)
3	ZeroAccess (5%)	NGRBot (2%)	Bagle (1%)	OpenCandy (0.07%)	NGRBot (1%)
4	SpyEye (4%)	Gigabid (2%)	Sality (.5%)	Unknown Adware.B (0.06%)	Toggle Adware (0.5%)
5	Windows Custodian (4%)	ZeroAccess (2%)	TDSS (.1%)	ZeroAccess (0.01%)	ZeroAccess (0.3%)
6	Karagany (4%)	Emit (1%)	Emit (.03%)	Whitesmoke (0.01%)	Gigabid (0.2%)
Total	32 families	19 families	6 families	6 families	40 families

Table 3: Most prominent families per infection vector by number of distinct binaries in each family, along with total families per vector.

Statistic	*Count*
Initial Driveby URLs	77,353
Initial Driveby Domains	16,302
Final Driveby URLs	13,998
Final Driveby Domains	6,309
Distinct Binaries from Drivebys	10,308

Table 4: Breakdown of driveby download datasets.

4.2 Components of a Driveby Download

Delving into our driveby malware, we examine how victims reach driveby downloads. For each binary that a driveby installs, we know the initial URL that directed a visitor to an exploit, the final URL that hosted the malware, and the family and monetization of the malware installed. Table 4 provides a summary of our dataset. In total, we receive 4,967 distinct binaries from Google Safe Browsing (as determined by MD5 sum) that are directed to by 77,353 initial URLs. In addition, we have 5,341 binaries installed by Blackhole and Phoenix that we use to understand malware families installed.[2]

We manually analyzed a sample of initial URLs and determined that they represent compromised legitimate sites, while we find that final URLs reflect dedicated infection sites hosting malware. (We provide further evidence of these observations in Section 4.4.) Examining the complexity of the infection chain, we find that 99.8% of initial pages lead to an exploit hosted on a different domain. We find over 46% of final domains are linked to by more than one initial URL, indicating driveby controllers infect multiple sites in order to maximize traffic while an exploit domain is alive. (Clearly, this fan-in could be much higher, since we may not have all initial URLs in our data-gathering.) Due to takedown and blacklisting of final driveby domains, 43% of initial domains redirect to more than one exploit domain. This may also be the result of demultiplexing traffic from a single initial URL to multiple miscreants willing to pay for traffic to their exploit page. Finally, 23% of final domains installed more than one family of malware.

4.3 The Role of Exploit Packs

4.3.1 Identifying Exploit Packs

We determine the exploit pack associated with a driveby download from Google's Safe Browsing list using similar techniques to those described in Section 3.3; in particular, automated clustering and manual analysis. We use a combination of three sources of ground truth to identify kits: access to exploit pack code for Blackhole, Eleonore, Phoenix, and Crimepack (typically obfuscated PHP); traces of traffic from live instances of other exploit kits; and community knowledge.

[2]It is important to note that when we analyze URL breakdowns and exploit kit popularity, we only use the Safe Browsing binaries, to avoid biasing our data towards specific exploit kits. In particular, we do not use the Blackhole/Phoenix feed, which would of course provide a bias towards those exploit kits.

Ranking	*Pack*	*Initial*		*Final*	
1	Blackhole	4,828	28%	1,075	29%
2	Incognito	1,779	10%	504	13%
3	Unknown.1	699	4%	74	2%
4	Sakura	393	2%	122	3%
5	Crimepack	219	1%	17	<1%
6	Unknown.2	200	1%	48	1%
7	Bleeding Life	188	1%	20	<1%
8	Phoenix	164	<1%	73	2%
9	Eleonore	34	<1%	14	<1%
–	Executable	8,186	49%	1,656	45%

Table 5: Popular exploit packs and the number of initial and final domains that lead to each. Executable indicates that the final URL ended with ".exe". To date we have been unable to attribute #3 and #6 on the list to a known exploit pack.

For all final URLs, we automatically generate clusters based on the query parameters embedded in each URL. An analyst then applies the appropriate label to each cluster. For instance, we know that a URL with parameters matching `w.php?f=(.*?)&e=(.*?)` belongs to Blackhole. We find these patterns are consistent across distinct domains due to the software-as-a-service nature of exploit kits; a single miscreant authors a kit and then sells it to multiple parties whom each run it on their own domains. As a result, major variations between URLs only appear in the domains, not the format of the URLs. For rare instances when we cannot identify a cluster, we provide an arbitrary label. In total, we identify 10 distinct clusters of exploit packs containing 92% of all URLs from the Google Safe Browsing list. We could not cluster the remaining 8% of URLs.

4.3.2 Exploit Kit Popularity

Table 5 shows a breakdown of the most popular exploit kits. Over 47% of initial domains terminate at an exploit kit, indicating the driveby ecosystem is permeated by exploit packs. Blackhole has the highest popularity, accounting for 28% of initial domains that led to an exploit. Incognito follows in popularity, along with a short list of other kits. Of the top 9, we could not identify two clusters. The remaining 49% of initial domains are found in URLs that link directly to malware executables. While these requests certainly reflect a successful exploit, we are unable to identify if an exploit pack was responsible.

4.3.3 Malware Families by Exploit Pack

Exploit kits drive the exploit-as-a-service ecosystem, allowing any miscreant to infect victims with their flavor of malware. We find that 77% of final domains that rely on exploit kits serve only a

Family	Executable	Blackhole	Incognito	Unknown.1	Sakura	Crimepack	Unknown.2	Bleeding Life	Phoenix	Eleonore
Zero Access	✓	✓	–	✓	✓	–	–	–	–	–
Windows Custodian	✓	–	–	–	–	–	–	–	–	–
Karagany	✓	✓	–	–	–	–	–	–	–	–
Spyeye	✓	✓	✓	–	–	–	–	–	–	–
TDSS	✓	–	–	✓	–	–	–	–	–	–
Cluster A	✓	✓	–	✓	–	–	–	–	✓	–
Zbot	✓	✓	–	–	–	–	–	–	✓	–
Multi Installer	–	–	–	–	–	–	–	–	–	–
Medfos	–	✓	✓	✓	✓	–	–	–	–	–
Cluster B	–	✓	–	✓	–	–	–	–	–	–
clickpotato	✓	–	–	–	–	–	–	–	–	–
Perfect Keylogger	✓	–	–	–	–	–	–	–	–	–
Emit	✓	–	–	–	–	–	–	–	–	–
Sality	–	–	–	–	–	–	–	–	–	–
Votwup	–	–	–	–	–	–	–	✓	–	–
Fake Rena	–	✓	–	–	✓	–	–	–	✓	–
Cluster C	✓	✓	–	–	–	–	–	–	–	–

Table 6: Malware families installed by different exploit packs, ordered by popularity (Section 4.4). Sality and Multi Installer were installed by a unique set of URLs that we could not attribute to an exploit pack.

single family of malware. If we aggregate these families across exploit kits, the multiple actors within the driveby ecosystem all using the same kit become apparent. Table 6 shows a mapping between exploit kits and the most popular malware families in our dataset. A checkmark indicates that a family was seen being distributed via a particular kit.

We find that Blackhole is far and away used to install the most variants of malware, indicating a wide range of malware authors rely on Blackhole's services. After Blackhole comes the as-yet-unidentified kit Unknown.1, along with Phoenix. Given that anyone can purchase a kit, we find it surprising to see Incognito and Bleeding Life used predominantly to distribute a single malware family. In particular, 92% of the domains leading to Incognito delivered Spyeye, an information stealer, while 59% of the installs performed by Bleeding Life were Votwup, a DoS bot. For three exploit packs, Crimepack, Unknown.2, and Eleonore, despite their relative popularity in Table 5, we do not observe them dropping popular families of malware, and in general we only find a few distinct MD5s distributed by all three packs.

In summary, our findings illustrate the complex relationships in the exploit-as-a-service market where single exploit kits are used by multiple miscreants, each chasing their own profits. Even if a single exploit kit disappears, a multitude of kits can take its place.

4.4 Driveby Popularity & Duration

For the last component of our analysis, we examine the volume of traffic that initial and final domains in the driveby infection chain receive. We find that domains that host exploit kits remain active for a median of only 2.5 hours. These domains receive a limited volume of traffic from several compromised websites. From these traffic estimates, we provide a unique perspective on the most popular malware families installed by the exploit-as-a-service market, as well as the impact of Google Safe Browsing on the lifetime of driveby domains.

Statistic	*Value*
Average Daily DNS Lookups	726,631,152
Average Daily Unique Lookups	75,915,099
Average Daily Registered Domains	9,691,212
Total Compressed Size (3/1/12–4/20/12)	3.58 TB

Table 7: Daily statistics for DNS dataset spanning March 1, 2012 to April 20, 2012.

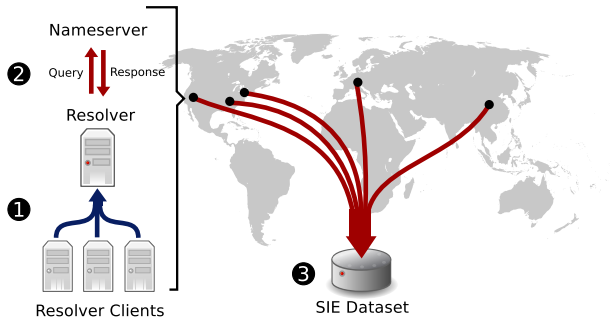


Figure 4: The SIE passive DNS data set contains DNS responses returned to recursive resolvers and authoritative servers. When a client attempts to resolve a domain, it contacts the recursive resolver configured by the ISP (❶); if the resolver does not have a cached entry for the domain, it contacts the relevant authoritative servers to resolve the domain for the client (❷); the resulting responses appear in the SIE passive DNS feed (❸).

4.4.1 Estimating DNS Traffic

To estimate the volume of traffic to driveby related domains, we obtained a dataset of passive DNS lookups recorded by the Security Information Exchange (SIE).[3] Our subset of the data consists of an average of 726 million daily DNS requests recorded from 135 distinct network vantage points over the course of March 1, 2012 to April 20, 2012. While the vantage points span numerous geographic locations and include a diversity of network environments, one US residential ISP heavily dominates the data, accounting for 75.6% of all lookups. As such, our analysis mainly characterizes users of this ISP, but the dataset also includes traffic from at least 9 other distinct networks providers spanning 18 cities and 3 countries. Table 7 summarizes the dataset; an average day includes 75 million unique DNS lookups to nearly 10 million registered domains. Identifying the 22,071 domains we know are tied to driveby downloads requires scanning through 3.58TB of compressed data in a MapReduce environment.

Figure 4 shows the process of how a DNS lookup appears in our dataset. When a client makes a DNS request, it first contacts its local resolver (❶). If the resolver has the answer in its cache, it replies immediately and does not generate a log entry in the SIE dataset. Conversely, upon a cache miss the resolver fetches the record from another resolver further up in the DNS hierarchy (❷). The SIE dataset contains all DNS responses returned to queries performed by participating resolvers; we do not see direct lookups from clients, nor does the dataset reveal the client IP addresses (❸). Each lookup contains the IP address of the DNS resolver, the outgoing query parameters, and the subsequent response, including domains, the reply's `TTL`, and associated `A` and `NS` records.

We must address a number of challenges to take the DNS lookups in this dataset and generate from it estimates of the total

[3] `https://sie.isc.org/`

volume of requests to driveby downloads (and, similarly, estimates of the number of clients affected). We define three sources of imprecision with the dataset:

Caching errors result from DNS caching at a resolver. When a client makes a lookup, we only see that lookup if it does not hit the resolver's cache. Furthermore, as TTLs differ across domains, a direct comparison between two domains without accounting for caching would lead to inaccuracy. Without accounting for caching errors, we would underestimate the volume of traffic to a domain.

Revisit errors occur for domains that the same client repeatedly queries over time. Revisit errors only affect estimates of the number of clients that request a domain. If we naively assume all clients only request a domain once, we will overestimate the number of clients accessing a driveby domain.

Unrelated query errors occur for compromised domains where only some of the domain's pages serve driveby downloads. For instance, if an attacker can only gain control of a specific subdirectory on a webserver, counting the number of DNS lookups to the domain will overestimate the volume of traffic flowing to a driveby exploit due to other legitimate traffic.

In order to account for caching errors, we adopt an approach described by Rajab et al. [32] to estimate the number of DNS lookups masked by caching. Specifically, DNS lookups are modeled as a Poisson process with query rate λ (if the query rates are not Poisson, this model introduces error, thus we emphasize our results should be used to rank the relative popularity of domains and not as exact estimates). We can estimate this rate from the interarrival times of valid DNS lookups. For a single resolver r, we calculate the interarrival time between two packets at time T_i and T_{i+1} as the difference between the end of the TTL window for the first request until the second request:

$$\Delta T_{r_i} = T_{r_{i+1}} - T_{r_i} - TTL \tag{2}$$

Given a series of DNS lookups D, the query rate λ can be approximated as:

$$\lambda_r \approx \frac{D}{\sum_{i=1}^{D} \Delta T_{r_i}} \tag{3}$$

Due to the additive nature of Poisson processes, estimating λ across all resolvers is simply the summation of individual λ_r estimates:

$$\lambda = \sum_{i=1}^{R} \lambda_r \tag{4}$$

We estimate the query rate λ for all of the domains in our dataset on a daily basis and project them to daily traffic estimates V by calculating the time between the first and last lookup for the day and the expected lookups per unit time:

$$V_{daily} = (T_{max} - T_{min}) \cdot \lambda_{daily} \tag{5}$$

For initial domains, many of which are compromised, this estimation approach is a requirement due to the frequency of DNS requests that can occur during a TTL cache window. The median interarrival time for requests for initial domains in our dataset is 4.5 minutes, 30% of which happen within one TTL window. For the final domains that host driveby exploits, these estimates prove unnecessary since the interarrival time of DNS lookups far exceeds the TTLs, leaving a small likelihood of a cache hit while the TTL is alive. The median TTL for final domains is only 30 seconds, while the median interarrival time of visits is 16 minutes. Coupling domain TTLs with their associated interarrival times, we find that fewer than 10% of queries happen within one TTL window.

Family	*Monetization*	*Volume*	*Feeds*
ZeroAccess	Dropper	35.0%	D,T,L
Windows Custodian	FakeAV	10.3%	-
Karagany	Dropper	9.5%	D
SpyEye	Info Stealer	8.0%	D
TDSS	Info Stealer	5.6%	A,L
Cluster A	Browser Hijacking	5.1%	-
Zbot	Info Stealer	5.0%	D
Multi Installer	Dropper	3.0%	-
Medfos	Browser Hijacking	2.6%	-
Cluster B	FakeAV	2.2%	-
Clickpotato	Adware	2.1%	D,L
Perfect Keylogger	Info Stealer	1.9%	L
Emit	Dropper	1.8%	D,A,L
Sality	Dropper	1.7%	A
Votwup	Denial of Service	1.6%	-
Fake Rena	FakeAV	1.5%	-
Cluster C	Info Stealer	0.7%	-

Table 8: Malware families ranked by traffic to the exploit domains hosting the family, along with whether the same family appeared in other feeds: (D)roppers, (A)ttachments, (T)orrents, and (L)ive.

Apart from caching errors, revisit errors and unrelated query errors have no simple solution. However, we find that most domains that host exploits are short-lived and not compromised, eliminating unrelated query errors that result from extraneous traffic (see Section 4.4.3). This leaves revisit errors, where the same client visiting a popular compromised site may make multiple DNS requests for domains hosting exploits. Servers that use "cloaking" to prevent clients from receiving an exploit more than once can lead us to overestimate the number of infected clients. As such, we only speak about exploit domains in terms of popularity and visits, not the number of successful infections.

4.4.2 Popularity of Malware Families Installed

Rather than ranking malware families by the number of variants (unique MD5s) as we did in Table 3, we use our estimates for the number of visits to driveby domains to determine the popularity of families installed. To start, for all of the final domains that launch an exploit, we estimate the total number of visits V each domain received during its lifetime, where V is the expected number of visits within a time period T given an estimated interarrival rate of DNS lookups independent of caching λ (explained in detail in Equation 5). We also know the set of exploit domains E that installed a particular malware family when the exploit domain was visited. Combined, we can then calculate the popularity of a family as:

$$\text{Popularity(family)} = \sum_{i \in E} V_i$$

Table 8 shows our results, with volume indicating the fraction of all DNS traffic to drivebys that belongs to a particular family. In total, we estimate that driveby exploit pages were queried 47,250 times over the course of 60 days. We find that 35% of these lookups are to domains hosting ZeroAccess, a dropper that can install a variety of other malware variants. This same family is also found in our dropper feed, torrents, and live traffic, indicating that miscreants rely on a multitude of infection vectors to propagate. The remainder of the list is dominated by fake anti-virus software, information stealers including SpyEye, and other droppers. Spam is noticeably absent from the top ranking, something mirrored in our other feeds.

4.4.3 Domain Duration

We estimate the uptime duration of a domain as the time between the first lookup that returns a valid `A` record until the time of the last valid lookup. Given the large size of the ISP predominant in our DNS data, such an estimate might not prove too far off from the true global duration of a domain, for instances of broadly targeted domains that will appear indiscriminately in the ISP's DNS traffic.

Figure 5 shows our results. Final domains are short lived, with a median duration of 2.5 hours. As such, parties searching for drive-bys require constant vigilance in order to detect the fast churn of malicious domains. In contrast, the initial domains that direct traffic to exploits have existed for a median of 44 days, with our data spanning a maximum of 60 days. This reiterates that many of the pages that funnel traffic to exploits are legitimate, compromised websites.

We perform the same duration analysis for domains contacted by binaries installed by driveby infections (denoted executions). Surprisingly, we find a similar behavior to initial domains. We manually analyze a sample of the connections that binaries make and find many domains belong to test connections or C&C infrastructure on compromised hosts. As such, estimating the population of clients infected with a family (not just visitors to those domains) based on DNS traffic (using the technique of Rajab et al. [32]) is wrought with revisit errors and unrelated query errors.

4.4.4 Popularity of Initial Domains

Apart from the final domains that host exploits, we also examine the relative popularity of websites that funnel traffic to drivebys. For ethical reasons, we do not reveal the names of popular compromised domains. Instead, we offer the volume of visitors they receive as a metric of their importance. We use our estimates of DNS lookups V, averaging lookups per domain. We find that a median site exhibits only 30 DNS lookups per day in our DNS dataset. However, 1% of initial domains receive over 22,548 estimated DNS lookups per day. To offer a comparison, in the dataset `google.com` receives an estimated 87,442 daily lookups. These results highlight that browser exploits are not relegated solely to obscure portions of the web, but include popular content that victims may regularly browse.

4.4.5 Safe Browsing Performance & Impact

The final metric we derive from DNS lookups concerns the impact of Google Safe Browsing on the lifetime of final domains hosting exploits. To perform the analysis, we draw upon a list of fine-grained timestamps corresponding to when Google identified each of the final domains in our dataset. We note that only 22% of final domains appear in the list, and these exhibit a bias towards longer living domains, with a median duration of 2 days compared to 2.5 hours.

From the passive DNS logs, we determine the first appearance of a valid DNS lookup to a domain $t_{\min}$ and the last valid lookup $t_{\max}$. We can then estimate the time before a malicious domain is identified by Google as $t_{\text{listed}} - t_{\min}$. Similarly, we can estimate the time a domain continues to receive traffic after it has been listed as $t_{\max} - t_{\text{listed}}$. Figure 6 shows a CDF of this calculation for all of the final domains in our dataset. A median malicious domain receives traffic for 17 hours before Google flags it, and continues to receive traffic for 11 hours after it is listed. These metrics reiterate the fast churn rate of driveby domains and the inherent challenge of tracking down newly created malicious sites.

The remaining 78% of final domains not flagged by Safe Browsing directly were identified only because of previously identified initial and intermediate URLs that lead to new final domains. As final domains tend to have short lifetimes, focusing on identifying initial URLs is more likely to help prevent users from visiting exploit sites.

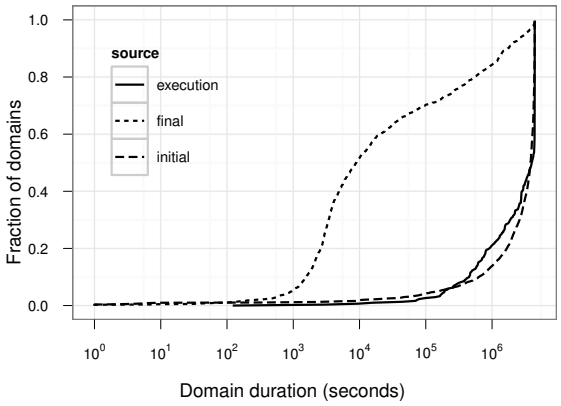


Figure 5: Length of time we see valid DNS requests and responses. This includes initial domains that funnel traffic, final domains which host exploits, and lastly domains contacted by malware once installed.

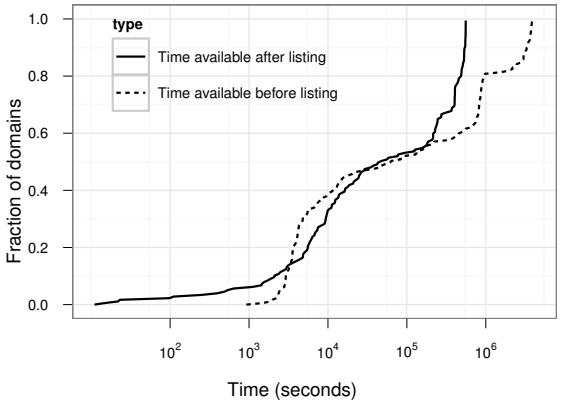


Figure 6: Time between miscreants creating a domain (first DNS lookup) until the domain is flagged by Google Safe Browsing, and subsequently, the time until the domain no longer receives traffic (e.g. unregistered, abandoned).

4.5 Discussion

We briefly discuss some of the important lessons we learned through our analysis and how we believe our findings should impact the development of new driveby detection technologies.

Malware diversity: Our malware feeds demonstrate that no single source of binaries is exhaustive; droppers, drivebys, email, warez, and live traffic all distribute binaries with various degrees of overlap. Furthermore, not all working malware samples produce network traffic, and those that do often contact compromised websites that simultaneously receive legitimate traffic, complicating network-based estimates of infections and clustering.

Crawler-based detection: The use of crawlers to identify malicious URLs often suffers from a few shortcomings, including the

challenge of identifying the continuous churn of short lived exploit domains and cloaking [30] employed by exploit kits. Future detection strategies would benefit from targeting the handful of exploit packs that exist, and in-browser defenses (deployed where detection must occur before a vulnerability is exploited) [22] remove some of the challenges surrounding cloaking and locating (not detecting) malicious websites.

Pay-per-install vs. exploit-as-a-service: Our prior work illustrated how the pay-per-install market plays a central role in the delivery of a diverse set of widespread malware families [4]. As we have shown, driveby downloads are also responsible for delivering popular droppers. We believe that while pay-per-install and drivebys might compete for some clients, they also serve complementary functions. By enabling PPI affiliates to use driveby downloads as an infection vector, PPI programs can aggregate installs across multiple exploit packs and compromised sites.

5. RELATED WORK

A number of studies have examined driveby downloads, seeking to identify them on the web and exploring both their hosting infrastructure and techniques to prevent exploits from succeeding.

Several studies have examined the prevalence of driveby downloads on the web. Moshchuk et al. [26] crawled 18 million URLs in an attempt to identify malicious content. They found that of 5.9% of the URLs crawled were driveby downloads, while 13.4% lead to spyware. A necessary component of profiling driveby downloads is a suitable detection engine. Curtsinger et al. [8] trained a classifier to detect heap spray exploits in JavaScript, and use the classifier to detect and blacklist malicious pages. Similarly, Cova et al. [7] use a modified browser to analyze web pages and extract features that can aid in identifying exploits that lead to driveby downloads. Their technique was further refined by Invernizzi et al. [15].

This work builds on our previous investigation into driveby downloads conducted by Provos et al. [29], where we examine the prevalence of such attacks and how exploited sites appear to users in search results and through syndicated advertisements. Provos et al. [29] also explored the hosting and distribution of driveby downloads in a very large dataset. Our study reflects an extension of this work by examining recent developments in exploitation and malware delivered by the exploits.

Polychronakis et al. examined malware distributed by driveby downloads, focusing on the command-and-control protocols used by malware once it executes [28]. Similarly, we use network characteristics as a feature to understand malware, but focus on using network and host-based metrics to identify malware families and monetization techniques.

A number of systems seek to prevent drivebys from injuring users [22, 47]. Our work differs from these since we focus on what happens before and after the driveby exploit has actually occurred. Our study has the potential to help guide the development of future such prevention and detection techniques.

6. SUMMARY

We examined the current landscape of driveby downloads, fueled both by the availability of exploit-as-a-service and compromised websites funneling legitimate users to malicious domains. Using two feeds of malicious URLs, we analyzed the malware that each site distributes via drivebys and the exploit kits and services employed to do so. We identified that nine exploit kits account for 92% of the malicious URLs in our dataset, 29% of which belong to Blackhole. These kits are used to distribute 32 of the most prominent families of malware, ranging from fake anti-virus to information stealers and browser hijacking, an indication that the exploit-as-a-service model plays an important role in the malware ecosystem.

Using passive DNS data, we identify that the infrastructure that hosts these exploit kits is short-lived and typically only receives traffic for 2.5 hours. We further explored the relative popularity of each kit and malware family, as well as the impact of Google Safe Browsing on the availability of malicious domains. Despite the pressures exerted by the security industry that cause rapid turnover in driveby domains, there is still ample need to improve the state of driveby detection and prevention. These solutions must account for the ease by which new driveby domains can appear, as well as their reliance on compromised websites for traffic.

7. ACKNOWLEDGEMENTS

We would like to thank the Arbor Networks ASERT Team for providing us with malware samples and VirusTotal for access to the thousands of virus scanner reports we used during classification.

This material is based upon work supported in part by the National Science Foundation under Grant No. CNS-0831535. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. This work is partially supported by the Office of Naval Research under MURI Grant No. N000140911081. This work is supported in part by the European Union through Grants FP7-ICT No. 256980 and FP7-PEOPLE-COFUND No. 229599. Juan Caballero is also partially supported by a Juan de la Cierva Fellowship from the Spanish Government. Parts of this work are supported by the Federal Ministry of Education and Research of Germany Grant 01BY1110, MoBE.

8. REFERENCES

[1] D. Anderson, C. Fleizach, S. Savage, and G. Voelker. Spamscatter: Characterizing internet scam hosting infrastructure. In *USENIX Security*, 2007.

[2] M. Bailey, J. Oberheide, J. Andersen, Z. Mao, F. Jahanian, and J. Nazario. Automated Classification and Analysis of Internet Malware. In *Proceedings of the Conference on Recent Advances in Intrusion Detection*, 2007.

[3] U. Bayer, P. Comparetti, C. Hlauschek, C. Kruegel, and E. Kirda. Scalable, Behavior-Based Malware Clustering. In *Network and Distributed System Security Symposium (NDSS)*, 2009.

[4] J. Caballero, C. Grier, C. Kreibich, and V. Paxson. Measuring Pay-per-Install: The Commoditization of Malware Distribution. In *Proceedings of USENIX Security*, 2011.

[5] J. Canto, M. Dacier, E. Kirda, and C. Leita. Large Scale Malware Collection: Lessons Learned. In *Workshop on Sharing Field Data and Experiment Measurements on Resilience of Distributed Computing Systems*, 2008.

[6] C. Y. Cho, J. Caballero, C. Grier, V. Paxson, and D. Song. Insights from the Inside: A View of Botnet Management from Infiltration. In *Proceedings of the USENIX Workshop on Large-Scale Exploits and Emergent Threats*, April 2010.

[7] M. Cova, C. Kruegel, and G. Vigna. Detection and Analysis of Drive-by-Download Attacks and Malicious JavaScript Code. In *Proceedings of the 19th International Conference on World Wide Web*, 2010.

[8] C. Curtsinger, B. Livshits, B. Zorn, and C. Seifert. Zozzle: Low-overhead Mostly Static JavaScript Malware Detection. In *Proceedings of the Usenix Security Symposium*, 2011.

[9] T. Cymru. A Criminal Perpsective on Exploit Packs. `http://www.team-cymru.com/ReadingRoom/Whitepapers/2011/Criminal-Perspective-On-Exploit-Packs.pdf`, May 2011.

[10] J. Franklin, V. Paxson, A. Perrig, and S. Savage. An Inquiry into the Nature and Causes of the Wealth of Internet Miscreants. In *ACM Conference on Computer and Communications Security (CCS)*, 2007.

[11] H. Gao, J. Hu, C. Wilson, Z. Li, Y. Chen, and B. Zhao. Detecting and Characterizing Social Spam Campaigns. In *Proceedings of the Internet Measurement Conference (IMC)*, 2010.

[12] T. Holz, C. Gorecki, F. Freiling, and K. Rieck. Detection and Mitigation of Fast-Flux Service Networks. In *Proceedings of the 15th Annual Network and Distributed System Security Symposium*, 2008.

[13] F. Howard. Exploring the Blackhole Exploit Kit. `http://nakedsecurity.sophos.com/exploring-the-blackhole-exploit-kit-3/`, March 2012.

[14] W. Huang, C. Hsiao, and N. Lin. mysql.com hacked, infecting visitors with malware. `http://blog.armorize.com/2011/09/mysqlcom-hacked-infecting-visitors-with.html`, September 2011.

[15] L. Invernizzi, P. M. Comparetti, S. Benvenuti, C. Kruegel, M. Cova, and G. Vigna. EvilSeed: A Guided Approach to Finding Malicious Web Pages. In *IEEE Symposium on Security and Privacy*, 2012.

[16] J. John, A. Moshchuk, S. Gribble, and A. Krishnamurthy. Studying Spamming Botnets Using Botlab. In *Usenix Symposium on Networked Systems Design and Implementation (NSDI)*, 2009.

[17] J. Jones. The State of Web Exploit Kits. BlackHat Las Vegas, July 2012.

[18] L. Kharouni. SpyEye/ZeuS Toolkit v1.3.05 Beta. `http://blog.trendmicro.com/spyeyezeus-toolkit-v1-3-05-beta/`, January 2012.

[19] C. Kreibich, C. Kanich, K. Levchenko, B. Enright, G. Voelker, V. Paxson, and S. Savage. Spamcraft: An Inside Look At Spam Campaign Orchestration. In *USENIX Workshop on Large-Scale Exploits and Emergent Threats*, 2009.

[20] C. Kreibich, N. Weaver, C. Kanich, W. Cui, and V. Paxson. GQ: Practical Containment for Measuring Modern Malware Systems. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pages 397–412. ACM, 2011.

[21] K. Levchenko et al. Click Trajectories: End-to-End Analysis of the Spam Value Chain. In *IEEE Symposium on Security and Privacy*, pages 431–446. IEEE, 2011.

[22] L. Lu, V. Yegneswaran, P. Porras, and W. Lee. BLADE: An Attack-Agnostic Approach for Preventing Drive-By Malware Infections. In *Proc. of the ACM conference on Computer and Communications Security*, 2010.

[23] MalwareIntelligence. Inside Phoenix Exploit's Kit 2.8 mini version. `http://malwareint.blogspot.com.es/2011/10/inside-phoenix-exploits-kit-28-mini.html`, October 2011.

[24] Malware Domain List. `http://www.malwaredomainlist.com`, July 2012.

[25] B. Miller, P. Pearce, C. Grier, C. Kreibich, and V. Paxson. What's Clicking What? Techniques and Innovations of Today's Clickbots. *Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 164–183, 2011.

[26] A. Moshchuk, T. Bragin, S. D. Gribble, and H. M. Levy. A crawler-based study of spyware on the web. In *Proceedings of the 2006 Network and Distributed System Security Symposium (NDSS)*, February 2006.

[27] F. Paget. An Overview of Exploit Packs. `https://blogs.mcafee.com/mcafee-labs/an-overview-of-exploit-packs`, 2010.

[28] M. Polychronakis, P. Mavrommatis, and N. Provos. Ghost turns Zombie: Exploring the Life Cycle of Web-based Malware. In *Proceedings of the 1st USENIX Workshop on Large-scale Exploits and Emergent Threats*, pages 1–8. USENIX Association, 2008.

[29] N. Provos, P. Mavrommatis, M. A. Rajab, and F. Monrose. All Your iFRAMEs Point to Us. In *Proceedings of the 17th Usenix Security Symposium*, pages 1–15, July 2008.

[30] M. Rajab, L. Ballard, N. Jagpal, P. Mavrommatis, D. Nojiri, N. Provos, and L. Schmidt. Trends in Circumventing Web-Malware Detection. Technical report, Google, July 2011.

[31] M. Rajab, L. Ballard, P. Mavrommatis, N. Provos, and X. Zhao. The Nocebo Effect on the Web: An Analysis of Fake Anti-Virus Distribution. In *USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, 2010.

[32] M. Rajab, F. Monrose, A. Terzis, and N. Provos. Peeking Through the Cloud: DNS-Based Estimation and Its Applications. In *Applied Cryptography and Network Security*, pages 21–38, 2008.

[33] K. Rieck, T. Holz, C. Willems, P. Düssel, and P. Laskov. Learning and Classification of Malware Behavior. *Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 108–125, 2008.

[34] Robert Lemos. MPack developer on automated infection kit (interview). `http://www.theregister.co.uk/2007/07/23/mpack_developer_interview`, July 2007.

[35] C. Rossow, C. Dietrich, and H. Bos. Large-Scale Analysis of Malware Downloaders. In *Proceedings of 9th Conference on Detection of Intrusions and Malware & Vulnerability Assessment*, 2012.

[36] C. Rossow, C. Dietrich, H. Bos, L. Cavallaro, M. van Steen, F. Freiling, and N. Pohlmann. Sandnet: Network Traffic Analysis of Malicious Software. In *Proceedings of the First Workshop on Building Analysis Datasets and Gathering Experience Returns for Security*, pages 78–88. ACM, 2011.

[37] SellMeYourTraffic.com. `http://www.sellmeyourtraffic.com`, May 2012.

[38] URL Query. `http://urlquery.net/report.php?id=40035`, May 2012.

[39] B. Stone-Gross, R. Abman, R. Kemmerer, C. Kruegel, D. Steigerwald, and G. Vigna. The Underground Economy of Fake Antivirus Software. In *Proceedings of the Workshop on Economics of Information Security (WEIS)*, 2011.

[40] B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szydlowski, R. Kemmerer, C. Kruegel, and G. Vigna. Your Botnet is My Botnet: Analysis of a Botnet Takeover. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 635–647, 2009.

[41] B. Stone-Gross, T. Holz, G. Stringhini, and G. Vigna. The Underground Economy of Spam: A Botmaster's Perspective of Coordinating Large-Scale Spam Campaigns. In *USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, 2011.

[42] B. Stone-Gross, R. Stevens, A. Zarras, R. Kemmerer, C. Kruegel, and G. Vigna. Understanding Fraudulent Activities in Online Ad Exchanges. In *Proceedings of the Internet Measurement Conference (IMC)*, 2011.

[43] H. Suri. The BlackHole Theory. `http://www.symantec.com/connect/blogs/blackhole-theory`, February 2011.

[44] K. Thomas, C. Grier, D. Song, and V. Paxson. Suspended Accounts in Retrospect: An Analysis of Twitter Spam. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, 2011.

[45] Traffbiz: A new malicious twist on affiliate partnerka schemes? `http://nakedsecurity.sophos.com/2012/03/01/traffbiz-a-new-malicious-twist-on-affiliate-partnerka-schemes/`, May 2012.

[46] Trustwave Spider Labs. Catch Me If You Can, Trojan Banker Strikes Again. `http://blog.spiderlabs.com/2012/05/catch-me-if-you-can-trojan-banker-zeus-strikes-again-part-2-of-5-1.html`, May 2012.

[47] Y.-M. Wang, D. Beck, X. Jiang, R. Roussev, C. Verbowski, S. Chen, and S. King. Automated Web Patrol with Strider HoneyMonkeys: Finding Web Sites That Exploit Browser Vulnerabilities. In *Proceedings of the 2006 Network and Distributed System Security Symposium (NDSS)*, 2006.

[48] S. Yadav, A. Reddy, A. Reddy, and S. Ranjan. Detecting Algorithmically Generated Malicious Domain Names. In *Proceedings of the Internet Measurement Conference (IMC)*, 2010.