Driving in the Cloud: An Analysis of Drive-by Download Operations and Abuse Reporting

Antonio Nappa[1,2], M. Zubair Rafique[1], and Juan Caballero[1]

[1] IMDEA Software Institute
[2] Universidad Politécnica de Madrid
{antonio.nappa,zubair.rafique,juan.caballero}@imdea.org

Abstract. Drive-by downloads are the preferred distribution vector for many malware families. In the drive-by ecosystem many exploit servers run the same exploit kit and it is a challenge understanding whether the exploit server is part of a larger operation. In this paper we propose a technique to identify exploit servers managed by the same organization. We collect over time how exploit servers are configured and what malware they distribute, grouping servers with similar configurations into operations. Our operational analysis reveals that although individual exploit servers have a median lifetime of 16 hours, long-lived operations exist that operate for several months. To sustain long-lived operations miscreants are turning to the cloud, with 60% of the exploit servers hosted by specialized cloud hosting services. We also observe operations that distribute multiple malware families and that pay-per-install affiliate programs are managing exploit servers for their affiliates to convert traffic into installations. To understand how difficult is to take down exploit servers, we analyze the abuse reporting process and issue abuse reports for 19 long-lived servers. We describe the interaction with ISPs and hosting providers and monitor the result of the report. We find that 61% of the reports are not even acknowledged. On average an exploit server still lives for 4.3 days after a report.

1 Introduction

Drive-by downloads have become the preferred distribution vector for many malware families [4, 33]. A major contributing factor has been the proliferation of specialized underground services such as exploit kits and exploitation-as-a-service that make it easy for miscreants to build their own drive-by distribution infrastructure [4]. In this ecosystem many organizations license the same exploit kit, essentially running the same software in their exploit servers (upgrades are free for the duration of the license and promptly applied). This makes it challenging to identify which drive-by operation a exploit server belongs to. This is fundamental for understanding how many servers an operation uses, which operations are more prevalent, how long operations last, and for prioritizing takedown efforts and law enforcement investigations.

A drive-by operation is a group of exploit servers managed by the same organization, and used to distribute malware families the organization monetizes. An operation may distribute multiple malware families, e.g., for different monetization schemes. A

malware family may also be distributed by different operations. For example, malware kits such as zbot or spyeye are distributed by many organizations building their own botnets. And, pay-per-install (PPI) affiliate programs give each affiliate organization a customized version of the same malware to distribute [5].

In this paper, we propose a technique to identify exploit servers managed by the same organization, even when those exploit servers may be running the same software (i.e., exploit kit). Our technique enables reducing the large number of individual exploit servers discovered daily, to a smaller, more manageable, number of operations. Our intuition is that servers managed by the same organization are likely to share parts of their configuration. Thus, when we find two servers sharing configuration (e.g., pointed by the same domain, using similar URLs, or distributing the same malware) this is a strong indication of both being managed by the same organization. To collect the configuration information we track exploit servers over time and classify the malware they distribute. Our data collection has been running for 11 months and has tracked close to 500 exploit servers.

Our analysis reveals two types of drive-by operations. Two thirds of the operations use a single server and are short-lived. The other third of the operations use multiple servers to increase their lifetime. These multi-server operations have a median lifetime of 5.5 days and some live for several months, despite individual exploit servers living a median of 16 hours. Miscreants are able to run long-lived operations by relying on pools of exploit servers, replacing dead servers with clones. We also observe a few short-lived multi-server operations (lasting less than a day) that use over a dozen exploit servers in parallel to achieve a burst of installations. While most short-lived operations distribute a single malware family, we observe multi-server operations often distributing more than one. In addition, we identify two PPI affiliate programs (the *winwebsec* fake antivirus and the *zeroaccess* botnet) that manage exploit servers so that their affiliates can convert their traffic into installations, without investing in their own drive-by infrastructure.

We also analyze the hosting infrastructure. We find that to sustain long-lived multi-server operations, in the presence of increasing pressure from defenders, miscreants are turning to the cloud. Over 60% of the exploit servers belong to cloud hosting services. Long-lived operations are using pools of exploit servers, distributed among different countries and autonomous systems (ASes) for resiliency, replacing dead servers with clones. Miscreants are taking advantage of a booming cloud hosting services market where hosting is cheap, i.e., virtual private servers (VPS) start at $10 per month and dedicated servers at $60 [23]. These services are easy to contract (e.g., automated sign-up procedures requiring only a valid credit card) and short leases are available (e.g., daily billing) so that the investment loss if the exploit server is taken down can be less than a dollar. In this environment, cloud hosting providers have started reporting that 50% of their automated VPS subscriptions are being abused [25].

To understand how difficult is to take down exploit servers, we issue abuse reports for 19 long-lived servers. We analyze the abuse reporting process, as well as the interaction with the ISPs and hosting providers. We use our infrastructure to monitor the result of the report (i.e., whether the server is taken down). The results are disheartening. Over

61% of the reports do not produce a reply and the average life of a exploit server after it is reported is 4.3 days.

Our work reveals a growing problem for the take down of drive-by download operations. While miscreants enjoy a booming hosting market that enables them to set up new exploit servers quickly, defenders face a tough time reporting abuse due to uncooperative providers and inadequate business procedures. Takedown procedures need to be rethought. There is a need to raise the cost for miscreants of a server being taken down, monitor short-lived VPS subscriptions, and shift the focus to prosecuting the organizations that run the operations, as well as the organizations behind specialized underground services supporting the ecosystem.

Finally, this work has produced a dataset that includes the malware binaries we collected, the metadata of when and how it was collected, and the malware classification results. To foster further research we make our dataset available to other researchers [13].

Contributions:

- We propose a technique to identify drive-by operations by grouping exploit servers based on their configuration and the malware they distribute.
- We report on aspects of drive-by operations such as the number of servers they use, their hosting infrastructure, their lifetime, and the malware families they distribute.
- We analyze the abuse reporting procedure by sending reports on exploit servers.
- We build a dataset with the collected malware, their classification, and associated metadata. We make this dataset available to other researchers.

2 Background

Drive-by downloads are a popular malware distribution vector. To distribute its products over drive-by downloads a malware owner needs 3 items: exploitation software, servers, and traffic. To facilitate the process, 3 specialized services exist (Figure 1). A malware owner can license an exploit kit (host-it-yourself), rent a exploit server with the exploit kit installed (exploitation-as-a-service), or simply buy installs from a pay-per-install service that provides the exploit server and the traffic.

2.1 Roles

The exploit kit ecosystem has four main roles: *malware owner*, *exploit kit developer*, *exploit server owner*, and *exploit server manager*. Exploit kit developers offer a software kit including a set of exploits for different platforms (i.e., combination of browser, browser plugins, and OS), web pages to exploit visitors and drop files on their hosts, a database to store all information, and an administration panel to configure the functionality and provide installation statistics. Exploit kits are offered through two licensing models: host-it-yourself (HIY) and exploitation-as-a-service (EaaS). In both models access to the exploit kit (or server) is time-limited and clients obtain free software updates during this time. Also in both models the client provides the traffic as well as a

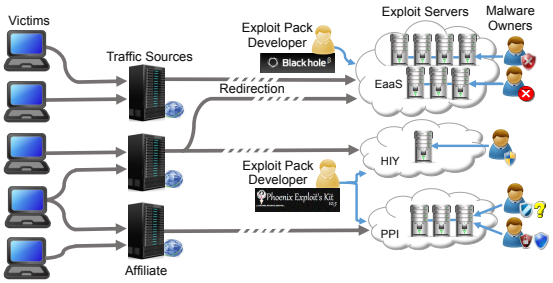

Fig. 1. Exploit kit ecosystem.

domain name to which the kit is linked. The client pays for domain changes (e.g., $20 for BlackHole [46]) unless it buys a more expensive multi-domain license.

The exploit server provider is the entity that contracts the hosting and Internet connectivity for the exploit server. It can be the malware owner in the HIY model or the exploit kit developer in EaaS. Exploit kits are designed to be installed on a single host that contains the exploits, malware files, configuration, and statistics. Thus, exploit servers are typically dedicated, rather than compromised, hosts. A robust hosting infrastructure is needed to launch long-lived operations as most exploit servers are short-lived. Exploit server providers acquire a pool of servers and favor hosting providers and ISPs where exploit servers live longer, i.e., those that are not diligent in handling abuse reports.

The exploit server manager is the entity that manages the exploit server through its administration panel. The manager is a client of the exploit kit developer and corresponds to the malware owner or a PPI service. PPI affiliate programs may run their own exploit server providing each affiliate with a unique affiliate URL. Affiliates credit installs by installing their affiliate-specific malware executable in hosts they have compromised, or by sending traffic to their affiliate URL, which would in turn install their affiliate-specific malware if exploitation succeeds. In these programs, affiliates can point their traffic sources to their affiliate URL in the program's exploit server or to their own exploit server. The latter requires investment but has two advantages: they can configure their exploit server to install other malware on the compromised machine, and they can avoid the affiliate program skimming part of their traffic for their own purposes. Our operation analysis reveals both exploit servers managed by individual affiliates and by PPI affiliate programs.

2.2 Shared Management

In this work we cluster exploit servers under the same management using information about the server's configuration. Two servers sharing configuration, (e.g., pointed by the same domain, using similar URLs, or distributing the same malware) indicates that they are managed by the same organization. We focus on server configuration because the software is identical in many exploit servers since kit updates are free and promptly

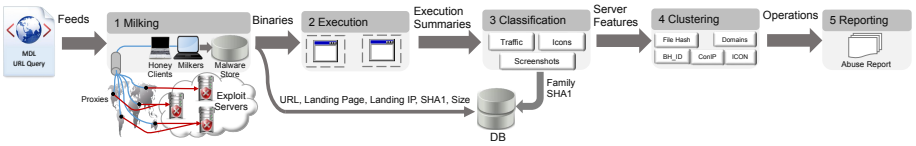

Fig. 2. Architecture of our milking, classification, and analysis.

applied (19 days after the launch of BlackHole 2.0 we could no longer find any live BlackHole 1.x servers). New exploit servers often reuse old configuration because the attacker simply clones an existing server, including its configuration.

Our clustering can be used by law enforcement during the pre-warrant (plain view) phase of a criminal investigation [42]. During this phase, criminal activity is monitored and targets of importance are selected among suspects. The goal of the plain view phase is gathering enough evidence to obtain a magistrate-issued warrant for the ISPs and hosting providers for the servers in the operation. Our clustering can identify large operations that use multiple servers, rank operations by importance, and help understanding whether they belong to individual owners or to distribution services.

3 Methodology

To collect the information needed to cluster servers into operations, we have built an infrastructure to track individual exploit servers over time, periodically collecting and classifying the malware they distribute. Our pipeline is described in Figure 2. We receive feeds of drive-by download URLs (Section 3.1), use honeyclients as well as specialized milkers to periodically collect the malware from the exploit servers those URLs direct to (Section 3.2), classify malware using icon information and behavioral reports obtained through execution in a contained environment (Section 3.3), store all information in a database, and use the collection and classification data for clustering exploit servers into operations (Section 4) and for abuse reporting (Section 5). An earlier version of the milking and classification components were used to collect the BlackHole/Phoenix feed in [4]. Since that work, we have upgraded those two components. This section describes their latest architecture, detailing the differences with [4].

3.1 Feeds

To identify exploit servers for the first time, we use two publicly available feeds: Malware Domain List (MDL) [24] and urlQuery [41]. MDL provides a public forum where contributors report and discuss malicious URLs. The reported URLs are manually checked by volunteers. Once verified they are published through their webpage and feeds. urlQuery is an automatic service that receives URLs submitted by analysts and publishes the results of visiting those URLs on their webpage. We periodically scan the webpages

of MDL and urlQuery for URLs matching our own regular expressions for the landing URLs of common exploit kits. The volume of URLs in urlQuery is much larger than in MDL, but the probability of finding a live exploit server is larger in MDL because URLs in urlQuery are not verified to be malicious and URLs long dead are often re-reported.

3.2 Milking

Our milking component differs from the one used to collect the BlackHole/Phoenix feed in [4] in that it identifies an exploit server by its *landing IP*, i.e., the IP address hosting the landing URL, which provides the functionality (typically some obfuscated JavaScript code) to select the appropriate exploits for the victim's platform. In [4] we identified exploit servers by the domain in their URLs. This was problematic because a large number of domains often resolve to the IP address of an exploit server. When the domains in the URLs known to us went down, our milking would consider the exploit server dead, even if it could still be reachable through other domains. Currently, if all domains in the landing URLs of a server stop resolving, the milking queries two passive DNS services [27,28] for alternative domains recently observed resolving to the exploit server. If no alternative domain is found, the milking continues using the landing IP.

In addition, our infrastructure now resolves the malicious domains periodically, which enables locating previously unknown exploit servers if the same domain is used to direct traffic to different exploit servers over time. This information is used in our clustering (Section 4). Using this separate resolution we discover an additional 69 servers not present in our feeds and another 30 servers before they appear in the feeds.

Another difference is that in [4] we relied exclusively on lightweight *specialized milkers*, i.e., custom HTTP clients that collect the malware from the exploit server, without running a browser or going through the exploitation process, simply by replaying a minimized network dialog of a successful exploitation. Our specialized milkers took advantage of the lack of replay protection in the BlackHole 1.x and Phoenix exploit kits. Since then we have added support for milking other exploit kits by adding *honeyclients*, i.e., Windows virtual machines installed with an unpatched browser (and browser plugins), which can be navigated to a given landing URL [26,43].

Milking policy. Our milking tries to download malware from each known exploit server every hour on average. If no malware is collected, it increments a failure counter for the exploit server. If a failure counter reaches a threshold of 6, the state of its exploit server is changed to offline. If malware is collected before 6 hours, its failure counter is reset. This allows milking to continue through temporary failures of the exploit server. In addition, the milking component runs a separate process that checks if an offline exploit server has resurrected every 2 days. If three consecutive resurrection checks fail, the exploit server is considered dead. If the server has resurrected, its failure and resurrection counters are reset.

Fig. 3. Icon polymorphism. Each pair of icons comes from two different files of the same family and is perceptually the same, although each icon has a different hash.

	Feature	Th.	Clus.	Precision	Recall	Time
I	avgHash	3	126	99.7%	91.3%	1.6s
I	pHash	13	135	99.8%	89.5%	47.5s
S	avgHash	1	60	99.1%	65.3%	7m32s
S	pHash	13	51	98.2%	67.2%	11m5s

Table 1. Clustering results for icons (top) and screenshots (bottom).

3.3 Classification

Our classification process leverages icon information extracted statically from the binary as well as network traffic and screenshots obtained by executing the malware in a contained environment. Compared to the classification process in [4], we propose the automated clustering of malware icons using perceptual hashing. In addition, we evaluate the accuracy of the icon and screenshot clustering using a manually generated ground truth.

Malware execution. We execute each binary in a virtualized environment designed to capture the network traffic the malware produces, and to take a screenshot of the guest VM at the end of the execution. We use Windows XP Service Pack 3 as the guest OS and only allow DNS traffic and HTTP connections to predefined benign sites to leave our contained environment. All other traffic is redirected to internal sinks.

Our classification applies automatic clustering techniques separately to the icons, the screenshots, and the network traffic. Then, an analyst manually refines the generic labels by comparing cluster behaviors against public reports. Finally, majority voting on the icon, screenshot, and network labels decides the family label for an executable.

Icons. A Windows executable can embed an icon in its header. Many malware families use icons because it makes them look benign and helps them establish a brand, which is important for some malware classes such as rogue software. Icons can be extracted statically from the binary without running the executable, so feature extraction is very efficient. A naive icon feature would simply compute the hash of the icon. However, some malware families use polymorphism to obfuscate the icons in their executables, so that two malware of the same family have icons that look the same to the viewer, but have different hashes (Figure 3). To capture such polymorphic icon variants we use a perceptual hash function [48]. Perceptual hash functions are designed to produce similar hashes for images that are perceptually (i.e., visually) similar. A good perceptual hash returns similar hashes for two images if one is a version of the other that has suffered transformations such as scaling, aspect ratio changes, or small changes in brightness, contrast, and color. We have experimented with two different perceptual hash functions: average hash (avgHash) [21] and pHash [48]. We use the Hamming distance between hashes as our distance metric. If the distance is less than a threshold both icons are clustered together using the aggressive algorithm in Section 4.2. We experimentally select the threshold value for each feature. Table 1 (top) shows the clustering results on 5,698 icons compared with the manually generated ground truth, which an analyst

produces by examining the clusters. The results show very good precision for both features and slightly better recall and runtime for avgHash.

Screenshots. The screenshot clustering also uses perceptual hashing. Table 1 (bottom) shows the clustering results on 9152 screenshots. This time avgHash achieves better precision but slightly worse recall. The lower recall compared to the icons is due to the perceptual hashing distinguishing error windows that include different text or the icon of the executable. Still, the clustering reduces 9152 screenshots to 50–60 clusters with very high precision, so it becomes easy for an analyst to manually label the clusters. We ignore clusters that capture generic error windows or do not provide family information, e.g., the Windows firewall prompting the user to allow some unspecified traffic.

Network traffic. Our network clustering uses the features in [4]. Once clustered, an analyst generates traffic signatures for the clusters, so that the next clustering only needs to run on samples that do not match existing signatures.

Overall, our classification produces traffic labels for 80% of the executables, icon labels for 54%, and screenshot labels for 22%. It classifies 93% of the executables, 4% fail to execute, and 3% remain unclassified.

4 Exploit Server Clustering

To identify exploit servers managed by the same organization we propose a clustering approach, which leverages features derived from our milk data that capture how exploit servers are configured.

4.1 Features

We define 5 boolean server similarity features:

1. *Landing URL feature:* The landing URL of a exploit server contains elements that are specific to the configuration of the exploit kit. In particular, the file path in the landing URL (the directory where the kit's files are installed and the name of those files) and the parameter values (typically used to differentiate traffic sources) are configurable and changed from the default by the manager to make it difficult to produce URL signatures for the kit. This feature first extracts for each landing URL the concatenation of the file path (including the file name) and the list of parameter values. The similarity is one if the set intersection is non-empty, otherwise it is zero.
2. *Domain feature:* If the same DNS domain has resolved to the IP addresses of two exploit servers, that is a strong indication that both exploit servers belong to the same organization, i.e., the one that owns the domain. This feature first extracts the set of DNS domains that have resolved to the IP address of each server. The similarity between two servers is one if the set intersection is non-empty, otherwise the similarity is zero.
3. *File hash feature:* A malware owner can distribute its malware using its own infrastructure (HIY or EaaS) or a PPI service. However, it is unlikely that it will use

both of them simultaneously because outsourcing distribution to a PPI service indicates a willingness to avoid investing in infrastructure. Thus, if the same malware executable (i.e., same SHA1 hash) is distributed by two servers, that is a strong indication of both exploit servers belonging to the same organization. This feature first extracts the set of file hashes milked from each exploit server. The similarity is one if the set intersection is non-empty, otherwise it is zero.

4. *Icon feature:* The icon in a malware executable is selected by the creator of the executable, i.e., malware owner or an affiliate PPI program (the program is typically in charge of repacking the affiliate-specific malware [5]). In both cases a shared icon in files distributed by different servers is a strong indication of both servers distributing malware from the same owner. This feature is related to the file hash feature but covers files that may have been repacked while keeping the same icon. This feature first extracts the set of icons in files milked from each exploit server. The similarity is one if the set intersection is larger than 1 otherwise it is zero.
5. *Family feature:* If two servers distribute the same malware family, and the malware family is neither a malware kit (e.g., zbot, spyeye) nor an affiliate program, then the two servers distribute malware of the same owner and thus share management. This feature is optional for the analyst to use because it requires a priori knowledge of which malware families are malware kits or affiliate programs, otherwise it may overcluster. This boolean feature first extracts the set of non-kit, non-affiliate malware families distributed by each exploit server. The similarity is one if the set intersection is non-empty, otherwise it is zero.

4.2 Clustering Algorithms

We experiment with two clustering algorithms: the partitioning around medoids (PAM) [20] and an aggressive clustering algorithm that groups any servers with some similarity.

Partitioning around medoids. The input to the PAM algorithm is a distance matrix. To compute this matrix we combine the server similarity features into a boolean server distance metric as $d(s_1, s_2) = 1 - (\bigvee_{i=1}^{5} f_i(s_1, s_2))$, where f_i is the server similarity feature i. Note that the features compute similarity (one is similar), while the distance computes dissimilarity (zero is similar). Once a distance matrix has been computed, we apply the PAM algorithm. Since PAM takes as input the number k of clusters to output, the clustering is run with different k values, selecting the one which maximizes the Dunn index [14], a measure of clustering quality.

Aggressive clustering. Our aggressive clustering first computes a boolean server similarity metric: two servers have similarity one if any of the server feature similarities is one (logical OR). Then, it iterates on the list of servers and checks if the current server is similar to any server already in a cluster. If the current server is only similar to servers in the same cluster, we add the server to that cluster. If it is similar to servers in multiple clusters, we merge those clusters and add the current server to the merged cluster. If it is not similar to any server already in the clusters, we create a new cluster for it. The complexity of this algorithm is $O(n^2)$, but since the number of servers is on the hundreds, the clustering terminates in a few seconds.

5 Reporting

Reporting abuse is an important part of fighting cybercrime, largely overlooked by the research community. In this section we briefly describe the abuse reporting process and the challenges an abuse reporter faces. In Section 6.5 we detail our experiences reporting exploit servers and discuss the current situation.

Five entities may be involved in reporting an exploit server: the *abuser*, the *reporter*, the *hoster* that owns the premises where the exploit server is installed, the *abuser's ISP* that provides Internet access to the exploit server, and *national agencies* such as CERTs and law enforcement. Sometimes, the ISP is also the hoster because it provides both hosting and Internet access to the exploit server. The abuser can also be the hoster if it runs the exploit server from its own premises.

The most common practice for reporting exploit servers (and many other abuses[3]), is to first email an abuse report to the ISP's abuse handling team, who will forward it to their customer (i.e., the hoster) if they do not provide the hosting themselves. If this step fails (e.g., no abuse contact found, email bounces, no action taken), the reporter may contact the CERT for the country where the exploit server is hosted or local law enforcement. There are two main reasons to notify first the abuser's ISP. First, in most cases a reporter does not know the abuser's or hoster's identity. But, the abuser's ISP is the entity that has been delegated the IP address of the exploit server, which can be found in the WHOIS databases [12]. Second, ISPs that are provided evidence of an abuse of their terms of service (ToS) or acceptable use policy (AUP) by a host unlikely to have been compromised (e.g., an exploit server), can take down the abusing server without opening themselves to litigation. This removes the need for law enforcement involvement, speeding the process of stopping the abuse.

Next, we describe 3 challenges a reporter faces when sending abuse reports.

Abuse report format and content. The Messaging Abuse Reporting Format (MARF) [16, 17, 37] defines the format and content for spam abuse reports. Unfortunately, it does not cover other types of abuse and proposals for extending it (e.g., X-ARF [45]) are still work-in-progress. In this work we use our own email template for reporting exploit servers. The key question is what information will convince an ISP of the abuse. The goal is to provide sufficient evidence to convince the ISP to start its own verification. The key evidence we include is a network trace of a honeyclient being exploited by the exploit server. We also include the IP address of the server, the first day we milked it, and pointers to public feeds listing the server.

Abuse contact address. Finding the correct abuse contact is not always easy (or possible). For spam, RFC 6650 states that abuse reports should only be sent to email addresses clearly intended to handle abuse reports such as those in WHOIS records or on a web site of the form abuse@domain [17]. Unfortunately, not all ISPs have an

[3] This practice also applies to other types of abuse such as C&C servers, hosts launching SSH and DoS attacks, and malware-infected machines. However, spam is commonly reported from a receiving mail provider to the sender mail provider and web server compromises are commonly first reported to the webmaster.

Malware executables milked	45,646
Unique executables milked	10,600
Domains milked	596
Servers milked	488
ASes hosting servers	236
Countries hosting servers	57
Malware executions	20,724
Total Uptime days	338

Fig. 4. Summary of milking operation.

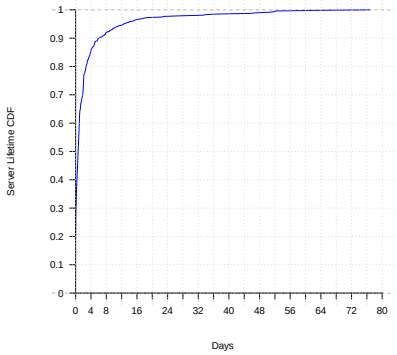

Fig. 5. CDF of exploit server lifetime.

abuse@domain address. Such addresses are only required for ISPs that (care to) have an abuse team [10] and have not been mandatory in WHOIS databases until recently. Even now, they are often only mandatory for new or updated WHOIS entries and the objects and attributes holding this information are not consistent across databases. We are able to find abuse addresses for 86% of all exploit servers we milk. In practice, reporters use WHOIS to identify the organization that has been delegated the abuser's IP address. If an abuse email does not exist for the organization (or cannot be found in its website) abuse reports are sent to the organization's technical contact, which is mandatory in WHOIS. Unfortunately, after finding an email address to send the report, there is no guarantee on its accuracy.

Sender's identity. Abuse reports may end up being received by malicious organizations (e.g., bullet-proof ISPs or hosters). Thus, using an individual's real identity in an abuse report can be problematic. On the other hand, abuse teams may be suspicious of pseudonyms. Organizations that issue many abuse reports such as SpamHaus [39] can rely on their reputation, but they do not act as abuse aggregators. In this work, we use a pseudonym to hide our identities and still get access to the communication with ISPs and hosters.

6 Analysis

Table 4 summarizes our milking, which started on March 7, 2012 and has been operating for 11 months (the BlackHole/Phoenix dataset in [4] covered only until April 20). We have milked a total of 488 exploit servers, hosted in 57 countries and 236 ASes, and downloaded from them 45,646 malware executables, of which 10,600 are unique (by SHA1 hash). A total of 596 DNS domains were observed pointing to the 488 servers.

ASN	Name	CC	Days up	ES	AS Rank Size	FIRE
16276	ovh	FR	192.62	20	805	10
701	uunet	US	100.62	1	3	-
44038	swisscom	CH	76.8	1	1,155	-
47869	netrouting	NL	70.0	18	6,537	-
43637	sol	AZ	61.1	1	12,676	-
48716	ps	KZ	52.0	1	25,772	-
56964	rmagazin	RO	49.5	2	21,273	-
12695	di-net	RU	47.6	9	478	-
36992	etisalat	EG	47.1	1	369	-
197145	infiumhost	RU	44.8	8	31,471	-
16265	leaseweb	NL	36.8	8	1,045	7
58182	kadroviy	RU	30.5	3	-	-
5577	root	LU	28.7	7	1,493	-
40676	psychz	US	28.1	5	6,467	-
21788	burst	US	27.8	14	1,344	-
28762	awax	RU	27.0	15	9,441	-
44784	sitek	UA	23.2	1	-	-
15971	ecosoft	RO	19.1	5	-	-

Table 2. Top ASes by cumulative exploitation time.

Family	Kit	ES	Files	Milk	Repack Rate
zbot	Kit	164	2,150	11,422	16.8
cridex		35	39	2,214	0.8
harebot		31	53	1078	1.5
winwebsec	Aff	18	5,820	16,335	59.5
zeroaccess	Aff	19	1,292	3,755	18.0
CLUSTER:A		9	14	266	2.2
spyeye	Kit	7	11	342	0.6
securityshield		5	150	307	11.8
CLUSTER:B		4	45	51	30.4
CLUSTER:C		4	1	4	1.0
smarthdd		4	68	453	3.1
CLUSTER:D		3	3	32	3.0
CLUSTER:E		3	1	4	1.0
CLUSTER:F		3	9	531	0.7
webprotect		3	3	26	3.9
cleaman		2	32	103	7.7
CLUSTER:G		2	5	148	1.5
CLUSTER:H		2	24	43	21.7
CLUSTER:I		2	9	17	9.4

Table 3. Malware family statistics.

6.1 Exploit Server Lifetime

To understand how well defenders are reacting to the drive-by download threat, we measure the exploit server lifetime, i.e., the period of time during which it distributes malware. For this measurement we use only exploit servers found after we updated our infrastructure to identify servers by landing IP (Section 3.2) and remove servers for which we have sent abuse reports (Section 6.5). Figure 5 presents the CDF for the exploit server lifetime. The majority of exploit servers are short-lived: 13% live only for an hour, 60% are dead before one day, and the median lifetime is 16 hours. However, it is worrying to observe a significant number of long-lived servers: 10% live more than a week, 5% more than two weeks, and some servers live up to 2.5 months.

The median exploit server lifetime we measure is more than six times larger than the 2.5 hours median lifetime of a exploit domain (a domain resolving to the landing IP of an exploit server) measured by Grier et al. using passive DNS data [4]. This shows the importance of identifying exploit servers by their IP address, accounting for multiple domains pointing to the same server over time.

6.2 Hosting

In this section we analyze the hosting infrastructure. We find that miscreants are abusing cloud hosting services. We also find, similar to prior work [38,40], autonomous systems hosting an inordinate number of exploit servers, compared to the size of their IP space.

Cloud hosting services. Using WHOIS we can first determine which organization has been delegated the IP address of an exploit server and then use web searches to determine if it offers cloud hosting services. Our results show that at least 60% of the exploit servers belong to cloud hosting services, predominantly to Virtual Private Server (VPS) providers that rent VMs where the renter gets root access. This number could be larger because ISPs do not always reveal in WHOIS whether an IP address has been delegated to a customer, who may be a hosting provider. This indicates that drive-by operations have already embraced the benefits of outsourcing infrastructure to the cloud.

AS distribution. Table 2 shows the top ASes by the cumulative uptime (in days) of all exploit servers we milked in the AS. It also shows the number of exploit servers in the AS, the CAIDA ranking of the AS by the number of IPv4 addresses in its customer cone (the lower the ranking the larger the AS) [6], and the FIRE ranking for malicious ASes [40]. The two ASes with the largest number of exploit servers are in Europe and the average life of an exploit server in those ASes is 10 days and 4 days respectively, well above the median lifetime of 16 hours. Some small ASes host an inordinate number of exploit servers compared to their ranking such as `awas` and `infiniumhost`, both located in Russia. There are also 3 ASes in Eastern Europe that do not advertise any IP addresses or no longer exist, which could indicate that they were set up for such operations. We milked servers in 3 ASes that appear in the 2009 FIRE ranking. Two of them (`ovh` and `leaseweb`) appear also among our top ASes, which indicates that practices at those ASes have not improved in 3 years.

6.3 Malware Families.

Our classification has identified a total of 55 families. Table 3 shows for the top families, whether the family is a known malware kit or affiliate program, the number of servers distributing the family, the number of unique files milked, the total number of binaries milked from that family, and its repacking rate. Overall, the most widely distributed families are information stealers (zbot, cridex, harebot, spyeye), PPI downloaders (zeroaccess), and rogue software (winwebsec, securityshield, webprotect, smarthdd). The family most milked was winwebsec, a fake antivirus affiliate program, while the one distributed through most servers was zbot, a malware kit for stealing credentials.

Figure 6 shows the distribution of malware families over time. While most families are distributed through short operations, there are a few families such as zeroaccess, zbot, and harebot, which have been distributed throughout most of our study.

Families with shared ownership. Since different malware families target different monetization mechanisms, malware owners may operate different families to maximize income from compromised hosts. There are 50 servers distributing multiple malware families. Nine servers distribute different malware families through the same landing URL, during the same period of time, and to the same countries, e.g., a visit from the US with no referer would drop family 1, another visit from the US a few minutes later family 2, and then again family 1. This indicates those families share ownership, as there is no way to separate the installs from the different families. Some families that manifest shared ownership are: harebot and cridex, CLUSTER:D and cleaman, and se-

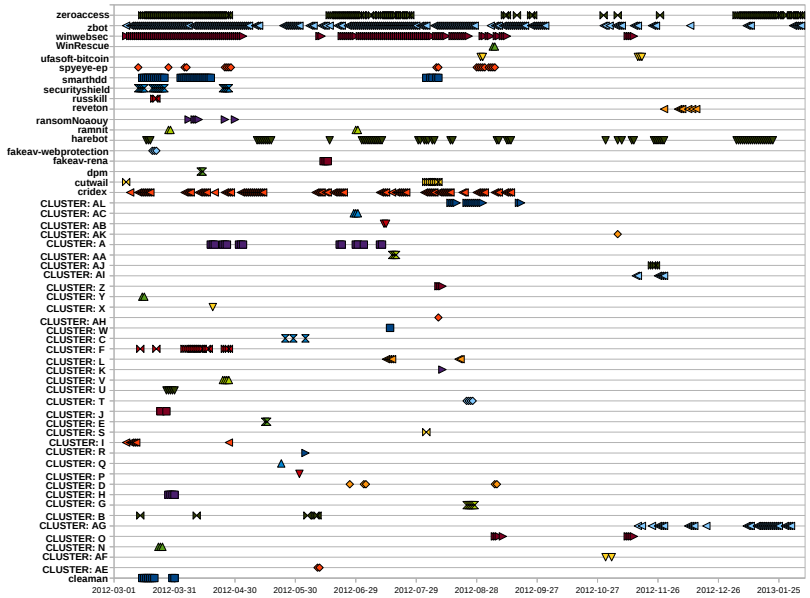

Fig. 6. Malware family distribution.

curityshield and smarthdd. There is also shared ownership involving families known to be malware kits or affiliate programs such as winwebsec affiliates installing zbot and CLUSTER:L, and zbot botmasters installing ramnit.

Repacking rate. Malware owners repack their programs periodically to avoid detection by signature-based AV. On average, a malware family (excluding kits and affiliate programs) is repacked 5.4 times a day in our dataset. This is a sharp rise compared to the 0.1 times a day prior work reported during August 2010 [5]. This trend will further harm the detection rate of signature-based AVs. The rightmost column in Table 3 shows the repacking rate for our top families. The rate for families known to be kits or affiliate programs is artificially high, covering multiple botnets or affiliates. There are other families with high repacking rates such as securityshield, CLUSTER:B and CLUSTER:H. This could indicate that those families are malware kits or affiliate programs.

6.4 Operations Analysis

In this section we evaluate our clustering approach to identify operations that use multiple exploit servers. Unfortunately, we do not have ground truth available to evaluate our clustering results in a quantitative fashion. In fact, if such ground truth was available, then there would be no need for the clustering. Instead, we argue qualitatively that our clustering identifies meaningful and interesting drive-by operations.

Table 4 summarizes the clustering results. We include the clustering results with and without the family feature for comparison. However, for the operation analysis below

	4 Features			5 Features		
Algorithm	**Clusters**	**Largest**	**Singletons**	**Clusters**	**Largest**	**Singletons**
Aggressive	172	64	119	108	127	70
PAM	256	31	188	204	31	141

Table 4. Clustering results.

we focus on the results without the family feature, since we suspect some families like securityshield to be affiliate programs. Since those are distributed alongside other malware, the family feature can overcluster. For each clustering algorithm the table shows the number of clusters, the size of the largest cluster, and the number of clusters with only one server. As expected, the aggressive algorithm groups the most, minimizing the number of clusters.

We first present a number of operations our clustering reveals (for the aggressive clustering with 4 features unless otherwise noted), evaluating their correctness with information not used by our features such as which kit was installed in the exploit server and for affiliate programs, which affiliate a malware executable belongs to (we extract the affiliate identifier from the network traffic). Finally, we summarize the types of operations the clustering reveals and their distribution properties including the number of servers used, their hosting, and the operation lifetime.

Phoenix operation. Using both PAM and aggressive all 21 Phoenix servers are grouped in the same cluster, which exclusively distributes zbot. Here, the clustering reveals that the Phoenix servers belong to the same operation *without* using any features about the exploit kit. Both algorithms do not include servers from other kits in the cluster, so they are not overclustering.

Reveton operation. We observe two clusters exclusively distributing the Reveton ransomware, which locks the computer with fake police advertisements. One cluster has 14 CoolExploit servers, the other 3 CoolExploit and one BlackHole 2.0. This agrees with external reports on the Reveton gang switching from BlackHole to the newer CoolExploit kit [34]. Here, the clustering captures an operation using different exploit kits, but possibly underclusters as both clusters likely belong to the same operation.

Winwebsec operation. We observe the winwebsec fake AV affiliate program distributed through 18 different servers in 8 clusters. There exists 3 singleton clusters, each exclusively distributing the winwebsec executable of a different affiliate. Another cluster of 8 servers distributes affiliate 60830 as well as another unknown malware family and zbot. The other 4 clusters distribute the executables of multiple affiliates. Here, there exist two possibilities: the same group could have signed up to the winwebsec program multiple times as different affiliates, or the affiliate program is managing the exploit server so that affiliates can convert their traffic into installs. To differentiate between both cases, we check their landing URLs. One of these clusters uses the same landing URL to distribute the executables of affiliates 66801, 66802, and 66803. In this case, there is no way to separate the installs due to each affiliate, which indicates those affiliates belong to the same entity. The other three clusters use different landing URLs for each affiliate, which indicates those servers are run by the affiliate program, which provides a distinct landing URL to each affiliate.

We confirm that the winwebsec program manages their own exploit servers through external means. We leverage a vulnerability on old versions of BlackHole, where the malware URLs used a file identifier that was incremented sequentially, and thus could be predicted. On March 12, we tried downloading file identifiers sequentially from one of the servers distributing multiple winwebsec affiliates. We found 114 distinct executables, of which 108 were winwebsec executables for different affiliates, one did not execute, and the other 5 corresponded to other malware families, including smarthdd and the Hands-up ransomware [47]. This indicates that on March 12, the winwebsec program had 108 affiliates and that the winwebsec managers, in addition to their own program, were also distributing other rogue software.

Zeroaccess operations. Zeroaccess is also an affiliate program [44]. With the aggressive algorithm there are 10 clusters distributing zeroaccess: 7 distribute a single affiliate identifier, the other 3 multiple. For two of these 3 the distribution is simultaneous and on a different landing URL for each affiliate, which indicates that the zeroaccess affiliate program also manages their own exploit server. The other distributes two affiliate identifiers on the same URL, indicating those affiliates belong to the same entity.

Zbot operations. There are 39 clusters distributing zbot in the aggressive clustering. Of these, 32 clusters distribute exclusively zbot, the largest using 21 servers over 6 days. For each of these 32 clusters we compute the set of C&C domains contacted by the malware milked from servers in the cluster. Only 3 of the 32 clusters have C&C overlap, which indicates that our non-family features capture enough shared configuration to differentiate operations distributing the same malware kit.

Broken malware operation. We identify a cluster with 13 servers that operates on a single day and distributes a single file. Surprisingly, the file does not execute. Apparently, the malware owners realized the malware was corrupt and stopped the operation.

Operations summary. The clustering reveals two types of operations. Two thirds of the clusters are singletons. They correspond to small operations with one server that lives on average 14 hours. Most singletons distribute a single family, which is often zbot or one of the generic families for which we have not found a published name. The remaining are operations that leverage multiple servers for their distribution. Multi-server operations use on average 6.2 servers and diversify their hosting. On average, each multi-server operation hosts 1.2 servers per country, and 2 servers per AS. Multi-server operations last longer with a median life of 5.5 days and only 1.2 servers operate on the same day. This indicates that they are replacing servers over time to sustain distribution, rather than using them for sudden bursts of installs (although we observe bursts like the broken malware operation mentioned earlier).

6.5 Reporting Analysis

We started sending abuse reports on September 3rd, 2012 for exploit servers that we had been milking for 24 hours. Most abuse reports did not produce any reply. Of the 19 reports we sent, we only received a reply in seven; 61% of the reports were not acknowledged. For two of the ISPs we were unable to locate an abuse@domain address

in WHOIS. One of these had no technical support contact either, so we resorted to web searches to find an email address. The absence of an abuse@domain address indicates a lack of interest in abuse reports. As expected, those reports did not produce a reply.

All initial replies contained a ticket number, to be included in further communications about the incident. Three of them also provided a URL for a ticket tracking system. Two of the replies came from ISPs to whom we had sent more than one report (on different dates). Surprisingly, only one of the two reports produced a reply. This lack of consistency indicates manual processing and that the response to an incident may depend on the abuse team member that first reviews the report.

After reporting a server, we keep milking it to understand how long it takes to act on a report. Note that, these reaction times are lower bounds because the servers could have been reported earlier by other parties. On average an exploit server lives 4.3 days after a report. Exploit servers whose report did not generate a response lived on average for 5.1 days after our report. Servers whose report produced a reply lived for 3.0 days. Thus, the probability of action being taken on the report when no reply is received is significantly smaller. Next, we detail the reactions to the 7 reports with replies.

Report 1. The most positive report. The exploit server was a VPS hosted by the ISP, which immediately disconnected it and notified us of the action (which we confirmed).

Report 2. This large US ISP replied with an automated email stating that they take abuse reports seriously but cannot investigate or respond to each of them. No further reply was received and the server lived for 4 days.

Report 3. A ticket was open with medium priority promising further notification. No further response was received and the server lived for another day.

Report 4. The report was forwarded to a customer. After a day the server was still alive so we sent a second report stating that the customer had not taken action and the ISP proceeded to disconnect the server.

Report 5. The report was forwarded to a customer and our ticket closed without waiting for the customer's action. The server was still alive for 1.7 days.

Report 6. The reply stated they would try to get back within 24 hours and definitely before 72 hours. The server lived two more hours and we never heard back.

Report 7. The initial reply stated that it was a customer's server and that according to the Dutch Notice and Take-down Code of Conduct [15], we had to notify the customer directly. Only if the customer did not reply after 5 days, or their reply was unsatisfactory, we could escalate it to them. We reported it to the client and after 5 days the server was still alive. We re-reported the exploit server to the ISP who told us to contact the customer again, which we did copying the ISP. This time the customer replied but was not willing to act on the response unless we reveal our real identity, which we declined. It seems that the ISP called them requesting the disconnection. The ISP later notified us about the disconnection. As far as we can tell, the five days waiting time is not part of the Dutch Notice and Take-down Code of Conduct.

These reports show that if the exploit server is hosted by a hosting provider who is a customer of the ISP, the ISP simply forwards them the abuse report and does no follow-up. It is up to the reporter to monitor the customer's action and re-report to the ISP in case of inaction. They also show how painful abuse reporting can be and the need for an homogeneous code of conduct for takedowns.

7 Related Work

A number of works have analyzed drive-by downloads. Wang et al. [43] build honeyclients to find websites that exploit browser vulnerabilities. Moshchuk et al. [26] use honeyclients to crawl over 18 million URLs, finding that 5.9% contained drive-by downloads. Provos et al. [33] describe a number of exploitation techniques used in drive-by downloads. They follow-up with a large-scale study on the prevalence of drive-by downloads finding that 67% of the malware distribution servers were in China [32]. Recently, Grier et al. [4] investigate the emergence of exploit kits and exploitation-as-a-service in the drive-by downloads ecosystem, showing that many of the most prominent malware families propagate through drive-by downloads. Our work differs from prior drive-by downloads analysis in that we focus on identifying and understanding the properties of drive-by operations, rather than individual exploit servers. Other work proposes detection techniques for drive-by downloads [9, 11, 49] and could be incorporated into our infrastructure.

Cho et al. [8], infiltrated the MegaD spam botnet and collected evidence on its infrastructure being managed by multiple botmasters. In contrast, our work shows how to automate the identification of servers with shared management, grouping them into operations. In simultaneous work, Canali et al. [7] analyze the security of shared hosting services. Similar to their work, we also issue abuse reports to hosting providers but our focus is on VPS services, which are more adequate for hosting exploit servers.

Prior works on running malware in a controlled environment have influenced our malware execution infrastructure [19, 22, 36]. Our classification builds on a number of prior works on behavioral classification techniques [1–5, 29, 31, 35] and incorporates the automated clustering of malware icons using perceptual hashing. We could also incorporate techniques to reduce the dimensionality in malware clustering [18] and to evaluate malware clustering results using AV labels [30].

8 Conclusion

We have proposed a technique to identify drive-by download operations by clustering exploit servers under the same management based on their configuration and the malware they distribute. Our analysis reveals that to sustain long-lived operations miscreants are turning to the cloud. We find that 60% of the exploit servers are hosted by specialized cloud hosting services. We have also analyzed the abuse reporting procedure with discouraging results: most abuse reports go unanswered and even when reported, it still takes several days to take down an exploit server.

9 Acknowledgements

The authors would like to thank Chris Grier and Kurt Thomas for their help and the anonymous reviewers for their insightful comments. This work was supported in part by the European Union through Grant FP7-ICT No. 256980 and by the Spanish Government through Grant TIN2012-39391-C04-01 and a Juan de la Cierva Fellowship for Juan Caballero. Opinions expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

References

1. D. S. Anderson, C. Fleizach, S. Savage, and G. M. 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 *RAID*, 2007.
3. U. Bayer, P. M. Comparetti, C. Hlauschek, C. Kruegel, and E. Kirda. Scalable, behavior-based malware clustering. In *NDSS*, 2009.
4. C. Grier et al. Manufacturing compromise: The emergence of exploit-as-a-service. In *CCS*, 2012.
5. J. Caballero, C. Grier, C. Kreibich, and V. Paxson. Measuring pay-per-install: The commoditization of malware distribution. In *USENIX Security*, 2011.
6. Caida. As ranking, 2012. `http://as-rank.caida.org`.
7. D. Canali, D. Balzarotti, and A. Francillon. The role of web hosting providers in detecting compromised websites. In *WWW*, 2013.
8. C. Y. Cho, J. Caballero, C. Grier, V. Paxson, and D. Song. Insights from the inside: A view of botnet management from infiltration. In *LEET*, 2010.
9. M. Cova, C. Kruegel, and G. Vigna. Detection and analysis of drive-by-download attacks and malicious javascript code. In *WWW*, 2010.
10. D. Crocker. Mailbox names for common services, roles and functions. RFC 2142, 1997.
11. C. Curtsinger, B. Livshits, B. Zorn, and C. Seifert. Zozzle: Low-overhead mostly static javascript malware detection. In *USENIX Security*, 2011.
12. L. Daigle. Whois protocol specification. RFC 3912, 2004.
13. Malicia project. `http://malicia-project.com/`.
14. J. C. Dunn. Well-separated clusters and optimal fuzzy partitions. *Journal of Cybernetics*, 4(1), 1974.
15. New dutch notice-and-take-down code raises questions, 2008. `http://www.edri.org/book/export/html/1619`.
16. J. Falk. Complaint feedback loop operational recommendations. RFC 6449, 2011.
17. J. Falk and M. Kucherawy. Creation and use of email feedback reports: An applicability statement for the abuse reporting format (arf). RFC 6650, 2012.
18. J. Jang, D. Brumley, and S. Venkataraman. Bitshred: Feature hashing malware for scalable triage and semantic analysis. In *CCS*, 2011.
19. J. P. John, A. Moshchuk, S. D. Gribble, and A. Krishnamurthy. Studying spamming botnets using Botlab. In *NSDI*, 2009.
20. L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*, volume 4. Wiley-Interscience, 1990.
21. N. Krawetz. Average perceptual hash, 2011. `http://www.hackerfactor.com/blog/index.php?/archives/432-Looks-Like-It.html`.
22. C. Kreibich, N. Weaver, C. Kanich, W. Cui, and V. Paxson. GQ: Practical containment for measuring modern malware systems. In *IMC*, 2011.

23. Love vps. http://www.lovevps.com/.
24. Malware domain list. http://malwaredomainlist.com/.
25. T. Morrison. How hosting providers can battle fraudulent sign-ups, 2012. http://www.spamhaus.org/news/article/687/how-hosting-providers-can-battle-fraudulent-sign-ups.
26. A. Moshchuk, T. Bragin, S. D. Gribble, and H. M. Levy. A crawler-based study of spyware on the web. In *NDSS*, 2006.
27. Bfk: Passive dns replication. http://www.bfk.de/bfk_dnslogger.html.
28. Ssdsandbox. http://xml.ssdsandbox.net/dnslookup-dnsdb.
29. R. Perdisci, W. Lee, and N. Feamster. Behavioral clustering of http-based malware and signature generation using malicious network traces. In *NSDI*, 2010.
30. R. Perdisci and M. U. Vamo: Towards a fully automated malware clustering validity analysis. In *ACSAC*, 2012.
31. M. Polychronakis, P. Mavrommatis, and N. Provos. Ghost turns zombie: Exploring the life cycle of web-based malware. In *LEET*, 2008.
32. N. Provos, P. Mavrommatis, M. A. Rajab, and F. Monrose. All your iframes point to us. In *USENIX Security*, 2008.
33. N. Provos, D. McNamee, P. Mavrommatis, K. Wang, and N. Modadugu. The ghost in the browser: Analysis of Web-based malware. In *HotBots*, 2007.
34. Cool exploit kit - a new browser exploit pack. http://malware.dontneedcoffee.com/2012/10/newcoolek.html/.
35. K. Rieck, T. Holz, C. Willems, P. Düssel, and P. Laskov. Learning and classification of malware behavior. In *DIMVA*, 2008.
36. C. Rossow, C. J. Dietrich, H. Bos, L. Cavallaro, M. van Steen, F. C. Freiling, and N. Pohlmann. Sandnet: Network traffic analysis of malicious software. In *BADGERS*, 2011.
37. Y. Shafranovich, J. Levine, and M. Kucherawy. An extensible format for email feedback reports. RFC 5965, 2010. Updated by RFC 6650.
38. C. Shue, A. J. Kalafut, and M. Gupta. Abnormally malicious autonomous systems and their internet connectivity. *IEEE/ACM Transactions of Networking*, 20(1), 2012.
39. The spamhaus project, 2012. http://www.spamhaus.org/.
40. B. Stone-Gross, Christopher, Kruegel, K. Almeroth, A. Moser, and E. Kirda. Fire: Finding rogue networks. In *ACSAC*, 2009.
41. urlquery. http://urlquery.net/.
42. R. J. Walls, B. N. Levine, M. Liberatore, and C. Shields. Effective digital forensics research is investigator-centric. In *HotSec*, 2011.
43. 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 *NDSS*, 2006.
44. J. Wyke. The zeroaccess botnet: Mining and fraud for massive financial gain, 2012. http://www.sophos.com/en-us/why-sophos/our-people/technical-papers/zeroaccess-botnet.asp:x.
45. X-arf: Network abuse reporting 2.0. http://x-arf.org/.
46. Xylitol. Blackhole exploit kits update to v2.0, 2011. http://malware.dontneedcoffee.com/2012/09/blackhole2.0.html.
47. Xylitol. Tracking cyber crime: Hands up affiliate (ransomware), 2011. http://www.xylibox.com/2011/12/tracking-cyber-crime-affiliate.html.
48. C. Zauner. Implementation and benchmarking of perceptual image hash functions. Master's thesis, Upper Austria University of Applied Sciences, 2010.
49. J. Zhang, C. Seifert, J. W. Stokes, and W. Lee. Arrow: Generating signatures to detect drive-by downloads. In *WWW*, 2011.