COMPUTERS & SECURITY 69 (2017) 155–173

Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

Analyzing the ecosystem of malicious URL redirection through longitudinal observation from honeypots

Mitsuaki Akiyama [a,*], Takeshi Yagi [a], Takeshi Yada [a], Tatsuya Mori [b], Youki Kadobayashi [c]

[a] NTT Secure Platform Laboratories, Tokyo, Japan
[b] Waseda University, Tokyo, Japan
[c] Nara Institute of Science and Technology, Ikoma, Nara, Japan

ARTICLE INFO

Article history:
Available online 11 January 2017

Keywords:
Honeypot
Compromised website
URL redirection
Drive-by download
Domain generation algorithm

ABSTRACT

Today, websites are exposed to various threats that exploit their vulnerabilities. A compromised website will be used as a stepping-stone and will serve attackers' evil purposes. For instance, URL redirection mechanisms have been widely used as a means to perform web-based attacks covertly; i.e., an attacker injects a redirect code into a compromised website so that a victim who visits the site will be automatically navigated to a malware distribution site. Although many defense operations against malicious websites have been developed, we still encounter many active malicious websites today. As we will show in the paper, we infer that the reason is associated with the evolution of the *ecosystem of malicious redirection*.

Given this background, we aim to understand the evolution of the ecosystem through long-term measurement. To this end, we developed a honeypot-based monitoring system, which specializes in monitoring the behavior of URL redirections. We deployed the monitoring system across four years and collected more than 100K malicious redirect URLs, which were extracted from 776 distinct websites. Our chief findings can be summarized as follows: (1) Click-fraud has become another motivation for attackers to employ URL redirection, (2) The use of web-based domain generation algorithms (DGAs) has become popular as a means to increase the entropy of redirect URLs to thwart URL blacklisting, and (3) Both domain-flux and IP-flux are concurrently used for deploying the intermediate sites of redirect chains to ensure robustness of redirection.

Based on the results, we also present practical countermeasures against malicious URL redirections. Security/network operators can leverage useful information obtained from the honeypot-based monitoring system. For instance, they can disrupt infrastructures of web-based attack by taking down domain names extracted from the monitoring system. They can also collect web advertising/tracking IDs, which can be used to identify the criminals behind attacks.

* *Corresponding author.*
E-mail address: akiyama.mitsuaki@lab.ntt.co.jp (M. Akiyama).

http://dx.doi.org/10.1016/j.cose.2017.01.003

1. Introduction

Attack campaigns targeting websites, e.g., Beladen, Gumblar, and Nine-ball, successfully affected tens of thousands of websites in 2009 (Websense Security Labs, 2009). In addition to this mass compromising, we faced serious server-side vulnerabilities such as Heartbleed, ShellShock, and Poodle in 2014 (Durumeric et al., 2014). The above threats exploit websites to expose many of them to the risk of data tampering. If such data tampering is applied to a website, it can generate web-based attacks for accomplishing attacker's *purposes*. One such purpose of inflicting serious damage to victims is to compel a compromised website to serve as a stepping-stone of various web-based attacks, e.g., drive-by download exploits. Once a victim visits such a compromised website, he/she will be redirected to another website, which exploits the vulnerabilities of web browsers to automatically download and install malware. URL redirection is used as a fundamental *mechanism* to broadly collect web accesses across websites. In addition, injecting redirect codes into various compromised websites is one *strategy* to surreptitiously conduct attacks.

As many defense operations against malicious websites have been developed (Grier et al., 2012; Moshchuk et al., 2006; Provos et al., 2008), the web-based attack, which consists of the aforementioned purposes, mechanisms, and strategies, must also have substantially evolved. We infer the reason we still encounter a large number of active malicious websites used for web-based attacks today may originate from the evolution of its core mechanism – *malicious URL redirection*.

On the basis of the discussion above, our goal in this study was to characterize the *ecosystem of malicious URL redirection*, which plays a vital role in the attack vector. While many new mechanisms have been incorporated into attacks, one interesting observation we found through the long-term study of the ecosystem was that URL redirection has always been used since the emergence of attacks. Such an *invariant* should play a key role in controlling the success of attacks.

With this background in mind, we pose the following research questions:

RQ1: *What are the key characteristics of URL redirection mechanisms?*
RQ2: *Have their purposes been changed over time?*

To answer the research questions, we developed a honeypot-based monitoring system, which specializes in monitoring the behavior of URL redirections. We deployed the monitoring system across four years, resulting in the collection of more than 100K malicious redirect URLs extracted from 776 websites compromised by fraudulent accesses with stolen credentials. We conducted an in-depth analysis of collected URL redirections.

Our key findings can be summarized as follows:

- Although the main purpose of URL redirection caused by redirect code injections has been drive-by download-based malware infection, click-fraud has become a new purpose in addition to malware infection in recent years.
- The use of *domain generation algorithm* (DGAs), which were originally used for communication of a bot and a command and control (C&C) server, has become popular as a means to increase the entropy of redirect URLs to thwart URL blacklisting.
- Both *domain-flux* and *IP-flux* should be concurrently used for deploying the intermediate sites of redirect chains to ensure robustness of redirection.

We unveiled the above change through four years of measurement. The insight obtained from analyzing the observed ecosystem helps network and security operators disrupt attack campaigns in appropriate points and layers corresponding to the distinct purpose, mechanism, and strategy. For this purpose, it is essential to reveal the purposes of attacks, mechanisms of redirection, and strategies to conduct attacks to gain security knowledge in a timely manner as well as protocol-level measurements such as passive DNS and web proxy. To the best of our knowledge, there have been no studies on the analysis of malicious websites introduced from the longitudinal viewpoint, i.e., even though some studies used a huge volume of datasets, these datasets were obtained by just one-time inspection or repeated inspections during a short period. Our study is a pioneer example of honeypot-based measurement to observe the ecosystem.

Based on the obtained knowledge, we also investigated the operational difference between types of DGAs and effectiveness of conventional methods, and discussed practical mitigation strategies against the ecosystem of malicious URL redirections: countering DGAs, discovering unknown compromised websites, and disabling web advertising and tracking IDs of attackers.

The rest of the paper is organized as follows. In Section 2, we detail our monitoring system and give a summary of the data we collected. In Section 3, we analyze the URL redirection mechanism in detail. In Section 4, we present an analyzed complex URL redirection structure, which uses DGAs. In Section 5, we discuss countermeasures to malicious URL redirection and domain-flux techniques. Then we evaluate the generality and impact of our observation in Section 6. We introduce related work in Section 7 and conclude this paper in Section 8.

2. Extracting URL redirection

2.1. Definition of URL redirection

Redirection refers to automatically replacing access destinations, and it is generally controlled over an HTTP protocol on the web. In addition to this conventional method, other methods for automatically accessing external web content, e.g., *iframe* tag, have been often used, particularly for web-based attacks. In this paper, we originally define that URL redirection additionally includes automatically occurring web access to URLs corresponding to an initial accessed URL and assume that URL redirection methods are tag redirections (`iframe`, `script`, `meta`, etc.), script redirections (JavaScript `location`'s methods), and HTTP redirection (HTTP-3xx status code).

2.2. Monitoring system

Discovering specific compromised websites in web space is required in advance of starting observation to understand the actual circumstances of URL redirect injection. Since web space is *extensive* (the number of URLs is increasing daily) and *deep* (dynamic pages are less easily indexed in search engines), it is not easy to discover malicious websites in the wild in a timely and efficient manner. Guided approaches for efficiently discovering unknown malicious websites in web space have been proposed (Invernizzi et al., 2012; Zhang et al., 2012). Additionally, a decoy system using honeypots to attract attackers has been proposed (Canali and Balzarotti, 2013). Our approach in conducting our research also involved honeypots to monitor compromised websites and analyzing malicious redirect code injected into them. We previously presented a prototype of our measurement system (Akiyama et al., 2013). In addition, we extend the scope of the above studies to continually track and dissect websites that are redirect destinations over the long term.

Our monitoring system is composed of several types of honeypots and can efficiently collect information of malicious URL redirection. The key method is purposely leaking the bait credentials, called *honeytokens* (Spitzner, 2003), of our web content management system (Web CMS), which is also a honeypot, to attackers to incur masquerade attacks. Fig. 1 gives an overview of our monitoring system, and we describe the analytical steps as follows:

1. We setup decoy Web Content Management Systems (CMSs) on the server, called Web CMS honeypot, and make it accessible from the open Internet by FTP protocol. We deploy famous CMSs such as Wordpress and Joomla. The CMSs contents, e.g., HTML, JavaScript, and PHP, with little modification are used as decoys and do not include external web content. The Web CMS honeypot has a domain name and requires a username and password to be able to access any content.
2. We setup a *malware sandbox*, running malware from our daily crawling of public websites and blacklisted websites i.e., websites listed in http://malwaredomainlist.com, in order to purposely leak honeytokens. The sandbox is designed to run malware which gathers usernames and passwords from the host system and leaks them to a machine elsewhere on the Internet under the control of an attacker. The sandbox generates honeytokens, i.e., IP address/domain name of the server, usernames and passwords of the CMSs created in the previous step, and sets them into a configuration file or registry of several client applications, e.g., FTP clients. In each analysis, the sandbox randomly generates unique credentials.
3. The Web CMS honeypot actually behaves as an FTP server and waits for attackers to infect the CMSs and inject URL redirection attack code and perhaps other things besides. It creates a user directory for each account corresponding to potentially leaked credential. After the attacker has successfully FTP login to the server with honeytokens, the attacker was provided with full access as an FTP user to the server, since the attacker could insert HTML tags, JavaScript code, PHP code and .htaccess files.
4. We run a honeyclient, which is a web browser-type honeypot, performing as a vulnerable web browser and access to each infected CMS content on a daily basis to determine where the URL redirections lead. In our observation, all observed redirections toward external websites must result from redirect code injection either directly or indirectly because original decoy content does not include external web content.

Our monitoring system restricts web access from outside benign web users to eliminate the risk of secondary attacks from our compromised websites, while it receives masquerade attacks and URL redirect injections. It means that our monitoring system permits only our honeyclient to access by web our compromised websites and publicly open FTP to incur masquerade attacks with stolen credentials.

2.3. Extraction method

In general, benign websites basically include benign redirects derived from original web content made by a website owner. Furthermore, an owner sometimes changes original web content legitimately. This type of redirect becomes noise in analyzing benign websites. In contrast, since all redirects on our

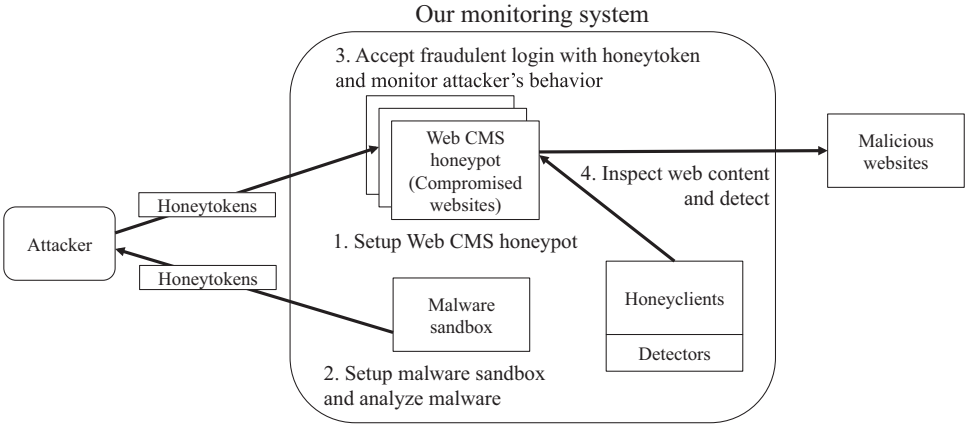

Fig. 1 – Monitoring system overview.

compromised *decoy* websites are caused by malicious redirect injections, we can safely assume that all the collected redirects are derived from malicious ones.

3. Analysis of URL redirection

To address **RQ1** and **RQ2**, we analyzed the methods and destination websites/URLs of injected URL redirection. First, we explain the basic statistics and give an overview of observed data. Next, the most significant security concern is how many web injections are related to malware infection, so we identify which redirect destinations are exploit sites. Then, we survey redirection methods and errors. Finally, we investigate domain and URL features such as URL popularity on individual websites.

3.1. *Basic statistics*

Table 1 summarizes the collected data. In our observation period, 776 websites were compromised in our monitoring system, and we confirmed that 96.5% of them have been actually injected with redirect codes over 57K times. Note that some redirect URLs directly use IP addresses for the `<host>` part of the URL instead of domain names, e.g., http://10.0.0.1/exploit.php. In consideration of this hostname variation, we simply express a `<host>` part as a name of a website in the following analysis and statistics. Emerging redirects indicate 11,235 websites of which 95.4% use domain names and the rest directly use IP addresses.

Table 1 – Summary of data.

Type	#
Period	39 months (Mar. 2012–May 2015)
Compromised websites	776 sites
Masquerader's login	59,462 logins
Content modification	57,009 logins
Inspections	323,581 times
Redirects (unique)	11,235 websites, 109,991 URLs

Table 2 – URL status when they were accessed.

Protocol	Status	# of accesses (%)
DNS/TCP	DNS resource not found	731,272 (49.7)
	TCP connection error	56,206 (3.8)
	Other error	51,732 (3.5)
HTTP	HTTP-2xx successful	453,143 (30.8)
	HTTP-3xx redirection	83,047 (5.6)
	HTTP-4xx client error	91,371 (6.2)
	HTTP-5xx server error	4,436 (0.3)
Total		1,471,197

In our experiment, there were 1.4M+ URL accesses derived from URL redirect injections on compromised websites across the four years of inspections. Table 2 shows the URL status of the accesses. Nearly half the accesses failed to establish a connection with websites due to DNS error or TCP connection error, while just about 30% of accesses successfully received an HTTP response. In total, we discuss the analysis of a large amount of accesses and clarify what they were aimed for, how they worked, and the reasons for access failure in the following sections.

Fig. 2 shows how actively events occurred in our monitoring system. It includes the numbers of masquerader's login events to compromised websites in our monitoring system, websites injected URL redirects, and redirect destination websites/URLs, which were calculated monthly. After that, the number of masquerade accesses reduced; however, the number of redirect destinations continued to remain. The main reasons for a number of redirect destinations remaining are caused by both web advertisements and automatically generated redirect destinations. We give detailed explanations on such redirect chains in Sections 3.4.4 and 4.

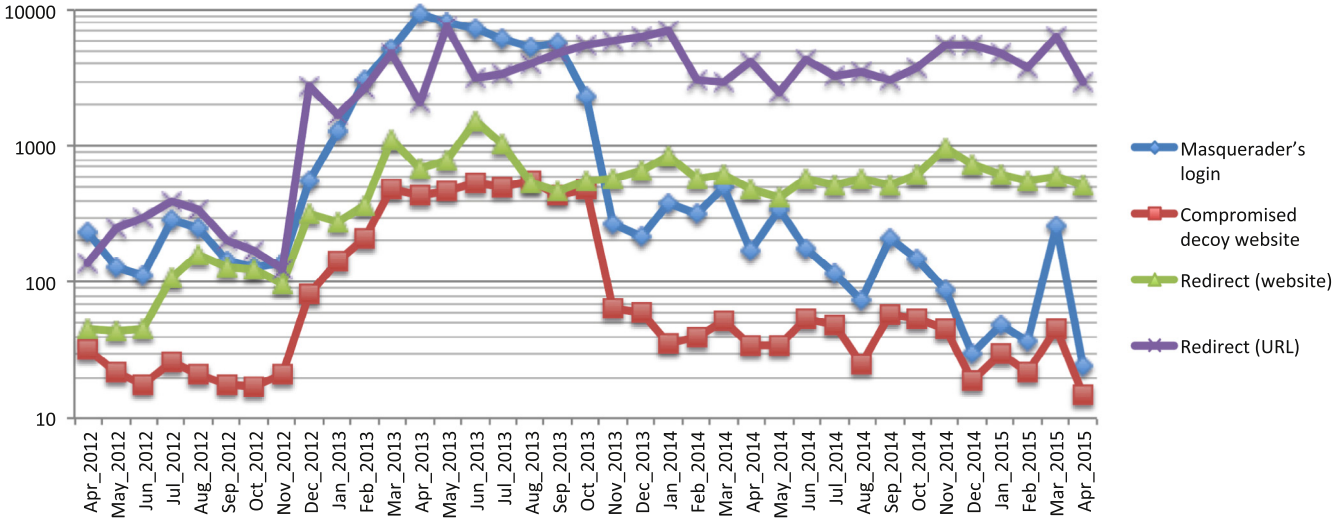

Fig. 2 – Injection activity.

3.2. *Malware-infection related websites*

3.2.1. *Detectors and detection coverage*

Due to the known limitation of detection coverages of static and dynamic analysis, we combine several detection methods that independently operate to broadly identify malicious websites related to malware infection. First, our honeyclient accesses compromised content and occasionally causes redirect destinations. Next, detectors analyze URLs or web content from the following viewpoints.

- **URL-based detector** statically parses URL strings by using YARA rules[1] embedded in Sandy (fb1h2s, 2014) and detects URLs based on the characteristic <path> part of a URL generated by known exploit-kits.
- **Content-based detector** statically scans web content, which is collected by our honeyclient, by using multiple types of anti-malware software[2] and detects URLs indicating scanned content when the content is flagged by at least more than one of the URLs.
- **Exploit-based detector** uses our honeyclient to dynamically analyze web content and detect malicious web content on the basis of anomalous browser/system behavior caused by browser exploit (i.e., drive-by download).

Our honeyclient has installed a vulnerable browser and its plugins and inspects an input URL with corresponding URLs that are automatically accessed using redirect methods. In the case of exploit-based detection, our honeyclient detects exploits on the basis of anomalous behavior in the system. To detect known vulnerabilities, the detection modules, called *HoneyPatches* (Akiyama et al., 2010), are applied to vulnerable functions on our honeyclient like a patch. They transparently check the data flow of vulnerable functions and detect exploits when input data triggers vulnerabilities.[3] To detect unknown vulnerabilities, our honeyclient also performs integrity checking of the file system, registry, and process, which are also implemented in Capture-HPC (Honeynet Project, 2008).

In addition, it extracts all redirect destinations including intermediate websites that have emerged during inspection.

A general limitation of a honeyclient is that false negatives are possibly raised by environment-dependent exploits in exploit-based detection. For example, to detect an exploit code targeting CVE-2010-0188, the environment of the honeyclient must first install a specific version of *Adobe reader*, which should be version 9.3.1 or earlier.

The URL-based detector works even if access fails because it only scans URL strings and detects characteristic URL-path patterns. Generally, exploit kits are widely used to build exploit sites, and these sites have characteristic URL paths since default templates for web content are used. Therefore, URL string signatures can be used to easily detect typical unchanged exploit kits.

URL access failure is out of the scope of detectors that use static or dynamic features of web content, i.e., content- and exploit-based detectors, while URL-based detectors are able to detect characteristic URLs. On the contrary, content-based and exploit-based detectors are able to detect URLs regardless of characteristic features in the URL-path part.

Table 3 – Detection overlap.

Sets	# of websites
U ∪ **C** ∪ **E** (All)	8311
U	7296
C	3694
E	1080
U ∩ **C**	2830
U ∩ **E**	788
C ∩ **E**	825
U – **C** – **E** (included only in **U**)	4350
C – **U** – **E** (included only in **C**)	711
E – **U** – **C** (included only in **E**)	139
U ∩ **C** ∩ **E**	672

U: URL-based detection; **C**: content-based detection; **E**: exploit-based detection.

3.2.2. *Detection and trend*

The detection overlaps between detectors are listed in Table 3. The URL-based detector identified 12,841 URLs owned by 7296 websites. The content-based detector identified web content corresponding to 18,273 URLs owned by 3694 websites. The exploit-based detector actually detected drive-by download exploits in 2841 inspections that included 6584 URLs owned by 1080 websites. Websites suspected of exploits were 74.1% (8333/11,235), and websites that definitely attempted to exploit were 9.61% (1080/11,235).

Although the URL-based detector identified a number of websites/URLs that appeared over an entire period, many such websites/URLs were access failures, and we did not obtain their actual web content. A number of websites identified by content-based and exploit-based detectors, i.e., websites related to drive-by downloads, appeared in 2012 and 2013 according to the sharp increase in the volume of attacker activity from the end of 2012 to Oct. 2013, as shown in Fig. 2. However, such drive-by downloads were not so active in 2014 and 2015 according to the drop in the volume of attacker activity.

3.2.3. *Correlative redirections in drive-by download sequence*

We analyzed how many URLs and websites are correlatively involved in drive-by download sequences (Fig. 3). We simply extracted all URLs and websites in each detected inspection. In inspection detecting drive-by downloads, all compromised websites have redirects leading to an exploit URL on an external website. The figure indicates that almost all sequences involved two or more external URLs and websites. This means that exploit codes are inevitably placed at external websites.

[1] The latest rules (Apr. 6, 2014) can be found at https://github.com/fb1h2s/sandy/tree/master/yara-ctypes/yara/rules/urlclassifier. To reduce obvious false positives, we excluded g01pack.yar, which aggressively detects benign URLs.

[2] Eleven distinct kinds of anti-malware software, that is, Avast, AVG, Avira, ClamAV, ESET, Forefront, Kaspersky, McAfee, Sophos, Symantec, and TrendMicro.

[3] Our HoneyPatch was presented in 2010 (Akiyama et al., 2010), and a similar but sophisticated implementation was presented by Araujo et al. (2014) in 2014.

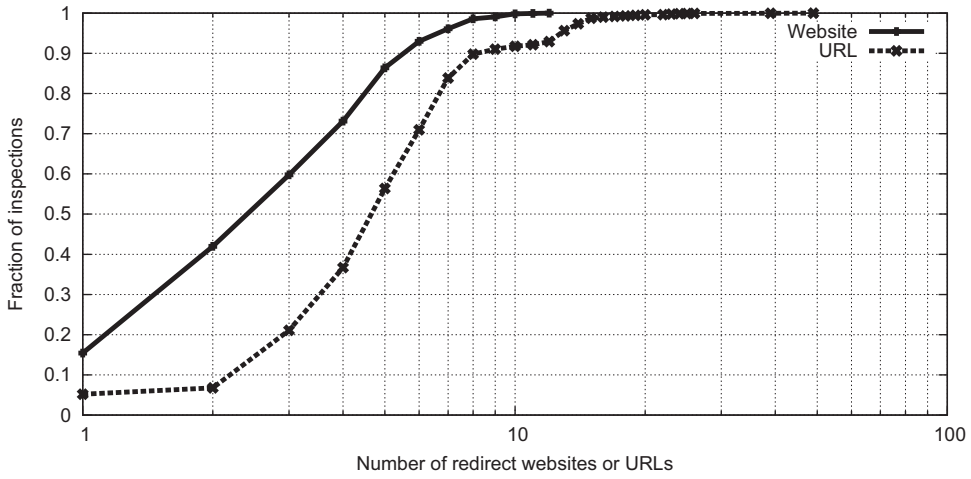

Fig. 3 – Redirection count in drive-by download exploits. Note that a landing URL that is on our compromised websites is excluded in this graph.

Exploit sites are built using various exploit kits, and typical exploit kits have several URLs that have different roles, such as browser fingerprinting, exploiting browser vulnerability, and forcing a compromised browser to download a malware executable. First, a page for browser fingerprinting identifies a target host's environment and redirects it to the appropriate exploit pages. If an exploit is successful, the target host is forced to access a malware download page.

The reason several inspections include two or more external websites is due to an *intermediate website*. In typical exploit sequences observed during our observation, an exploit site was accessed via an intermediate website. The role of this intermediate website is controlling web accesses and redirecting them to appropriate destinations. We explain the analysis of this intermediate website further in Section 4.2.

3.3. *Redirect features*

3.3.1. *Methods and utilization*

We classified the observed redirect methods into three categories: tag redirect, script redirect, and HTTP redirect. The most standard method is tag redirect using `script`, `iframe`, or `meta` tags. The `iframe` and `meta` tag insertions were only 1.0% and less than 0.1% respectively. In contrast, the `script` tag insertion became a major injection method over time, which accounted for 70.8% of all observed file modifications. In this case, almost all `script` tag insertions were `script` tags without *src* attribution. This means that obfuscated JavaScript is inside `script` tags and dynamically outputs `iframe` tags or executes `location` redirects after de-obfuscating itself. Due to this obfuscation, malicious redirect code conceals specific redirect destinations (i.e., URLs).

In comparison, HTTP redirects are implemented in server-side content as `header()` insertion in `.php` files (9 modifications) and `.htaccess` files (174 modifications). In the case of using the above redirect methods controlled in server-side content, it is impossible, by client-side measurement, to recognize the reason a redirection occurs, while a decoy website can collect such server-side `.php` and `.htaccess` files? Table 4 lists the classified redirect methods and the breakdown of their usage on actual injected redirect codes. We statically analyzed injected redirect codes, but this static analysis to find specific strings could not enable us to precisely identify D, E, and F because they are obfuscated and dynamically executed. Therefore, we counted `script` tag insertions without `src` attributes as the integrated number of D, E, and F.

3.3.2. *Errors*

Many redirections fail to access redirect destinations in some layers, e.g., DNS or HTTP. We counted the unique statuses of websites and indicated the time-series accessibility of redirect destinations in regard to websites, which are shown as

Table 4 – Redirect methods on compromised websites.

Category	Subcategory	Code example	# of modifications events
Tag redirect	A. iframe tag	iframe src = *URL*	574
	B. script tag	script src = *URL*	436
	C. meta tag	meta http-equiv = "Refresh" content = "0"; URL = *URL*	7
	D. iframe tag by `document.write`	document.write("¡iframe src = *URL*	39,949
	E. script tag by `document.write`	document.write("¡script src = *URL*	
Script redirect	F. JavaScript `location`	location.href = *URL* location.replace(*URL*)	
HTTP redirect	G. `PHP header()`	header("Location: *URL*	9
	H. .htaccess `RewriteRule`	RewriteRule ^.*$ *URL* [R = 302,L]	136
	I. .htaccess `ErrorDocument`	ErrorDocument 404 *URL*	14

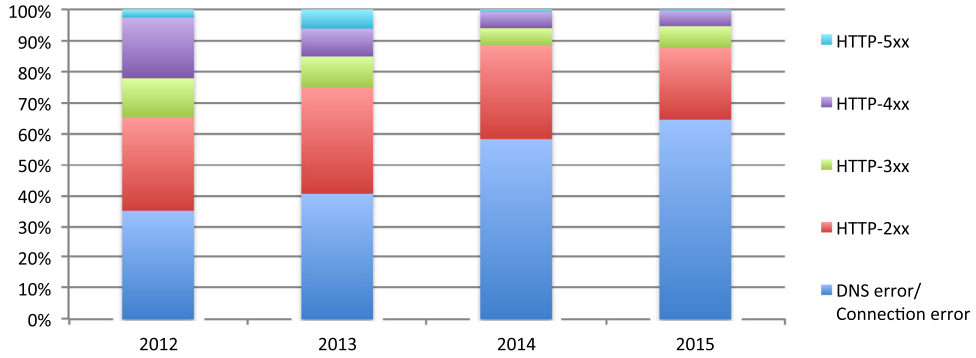

Fig. 4 – Time-series of access status.

percentages of DNS lookup failure, connection errors, and HTTP errors in Fig. 4. The percentages of DNS and TCP connection errors gradually increased.

During our observation period, 4488 domains responded with *non-existent domain* (NXDomain) more than once, and 32.9% of domains responded with NXDomain to all DNS queries. In addition, 20.7% of URLs responded with HTTP error (not HTTP-200) more than once, and 16.0% responded with HTTP error to all HTTP requests. These include automatically generated domains, and they often failed to resolve their domain name.

3.4. *Domain and URL features*

3.4.1. *Domain depth and suffix*

Table 5a shows the depth of domain, which means the number of labels in the `<host>` part of a URL, e.g., the domain depth of www.example.co.jp is 4. Domains placed 2nd and 3rd were dominate among all domains and accounted for 46.8 and 46.5%, respectively.

Table 5b lists the statistics for the (FQDN-1) label, which means domain suffix without a hostname, e.g., the (FQDN-1) label of www.example.com indicates example.com. Although mynumber.org and `.pro` are not popular domain suffixes, they are placed higher in rank. The reason of this unusual distribution of domain suffixes is that almost all domains under these two domain suffixes are *automatically generated domains* (AGDs), which are possibly produced using a DGA, and a vast majority are sometimes not accessible because of DNS lookup failure. Detailed analysis of the DGAs discovered during our observation is described in Section 4.

Table 5 – Domain depth and suffix statistics.

(a) Domain depth		(b) Distribution of domain suffix (Top 6)		
Domain depth	% domain	Domain suffix (FQDN-1)	Domain depth	% domain
2	46.8	.mynumber.org	3	17.95
3	46.5	.com	2	16.29
4	6.4	.pro	2	14.60
5	0.17	.ru	2	3.44
6	0.02	.metric.gstatic.com	4	2.87
7	0.01	.net	2	2.51

3.4.2. *Emerging redirect destinations from distinct compromised website*

Attackers frequently change redirect codes to switch redirect destinations to thwart blacklisting or change attack campaigns. Taking this feature into account, our monitoring system can efficiently obtain various redirect URLs through periodically monitoring compromised websites performing as a landing site of web-based attacks. In contrast to benign websites, decoy websites are suitable for long-term observation and can maximize the yield amount of redirect codes and redirect destinations. To indicate how many redirect destinations our monitoring system extracted from each compromised website, we counted observed URLs and websites derived from each compromised website over an entire measurement period (Fig. 5). The compromised websites had a median number of 91 domains and a median number of 127 URLs in terms of redirections in total.

3.4.3. *URL popularity on distinct redirect website*

From the viewpoint of redirect destination, Fig. 6 shows the URL popularity, which means *how many URLs each redirect website has*. A number of malicious websites, which are obviously exploit websites identified by the exploit-based detector mentioned in Section 3.2.1, are distributed between two and tens of URLs on the X-axis in the figure. A malicious website built using a certain exploit kit generally has several URLs (pages) such as a fingerprinting page for identifying a target's environment and an exploit page targeting specific vulnerabilities. Websites at the bottom right of this figure, which own a large volume of URLs, are used for web advertising and tracking, and we mention them in Section 3.4.4 in detail. The leftmost websites, which own only one URL, account for 69.3%, and most are the aforementioned AGDs. In the exploit-detected inspections mentioned in Section 3.2.2, these AGDs often emerged as intermediate websites; in other words, they fulfill the role of bridging an initial website, i.e., compromised website, and a final website, e.g., an exploit site. We further discuss the analysis of AGDs and unveil their nature in Section 4.

3.4.4. *Web advertising and tracking*

Even if an attacker compromises websites and injects redirect code, a redirect destination is not necessarily related to malware infection but rather web advertising and tracking

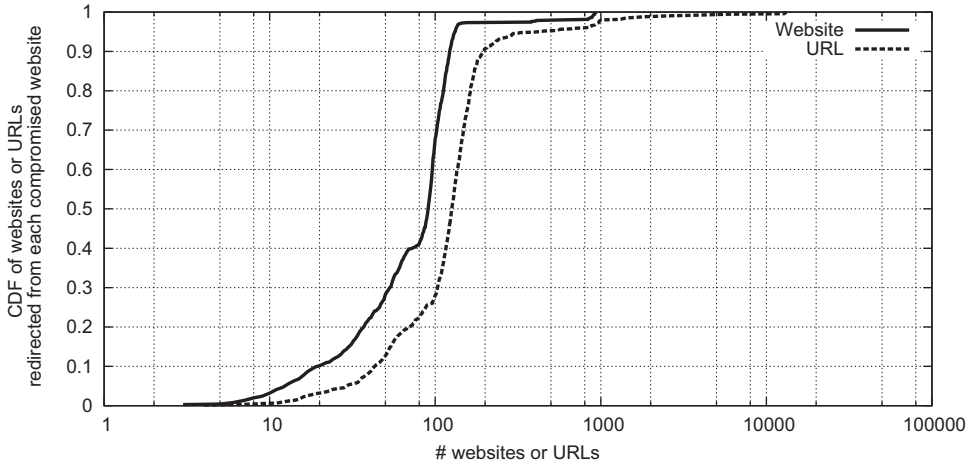

Fig. 5 – Number of emerging redirect destinations on distinct compromised website over entire observation period.

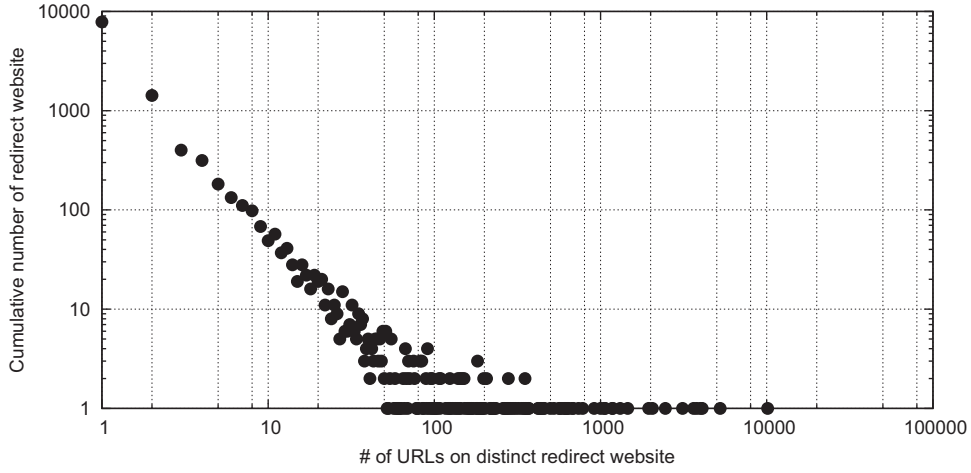

Fig. 6 – URL popularity on redirect website X-axis indicates cumulative number of URL on distinct redirect website through our observation. The top 10 and top 100 websites cover 36.7 and 67.9% of all observed URLs. Websites owning under 10 URLs are 93.9%; in particular, websites owning one URL are 69.1%.

in some cases. The aim of web advertising and tracking is to induce click-fraud monetization such as fraudulently utilizing pay-per-click advertising (PPC); therefore, this seems to be another aim of malicious URL redirection. An attacker formerly requires recruiting a large amount of malware-infected hosts, e.g., DNSChanger, Koobface, and ZeroAccess, to accomplish click-fraud monetization (Blizard and Livic, 2013). In contrast, injecting redirect code into compromised websites is an alternative to recruit general public web users for PPC. The more popular the website an attacker compromises, the more web users an attacker can simultaneously recruit without much effort. Therefore, injecting redirect code is more cost effective and scalable than recruiting click bots.

Websites having a large amount of URLs in our measurement (the bottom right of Fig. 6) are often used for web advertising or tracking. These URLs usually contain onetime-tokens or encoded individual data, e.g., timestamps, client information, and origin URLs (referrer URLs), in the <path> parts so that unique URLs are newly generated for every access. Table 6 lists the top websites owning a large amount of URLs. Only these 10 websites had 40,440 URLs, which accounts for 36.7% of all redirect URLs. We manually surveyed the usages of these websites and confirmed that almost all the websites are used for web advertising and tracking. To estimate how widely and commonly a certain URL is used, we calculate

Table 6 – Number of URLs on individual websites (Top 10).

Website	# of URLs	Prevalence (%)
user99[dot]freewebhostingarea[dot]com	10,127	0.6
googleads[dot]g[dot]doubleclick[dot]net	5,250	23.7
ad[dot]yieldmanager[dot]com	4,091	1.0
ads[dot]yahoo[dot]com	3,987	0.8
ib[dot]adnxs[dot]com	3,787	1.2
www[dot]google-analytics[dot]com	3,628	59.9
g[dot]adnxs[dot]com	3,115	1.0
pagead2[dot]googlesyndication[dot]com	2,454	28.4
s[dot]ad125m[dot]com	2,052	0.5
content[dot]yieldmanager [dot]edgesuite[dot]net	1,949	0.8

prevalence as shown in the table. This value is calculated by S_{emerge}/S_{exist}, where S_{exist} is the number of all websites, i.e., decoy compromised websites, and S_{emerge} is the number of websites whose redirect destination is set. While the domains of `doubleclick`, `google-analytics`, and `googlesyndication` provide popular web advertising and tracking services, they are also widely used by redirect codes injected into compromised websites. We mention the use of `google-analytics` for web tracking in Section 4.4.

The notable point about web advertising and tracking is that their numbers increased in 2014 and 2015 while the number of drive-by downloads gradually decreased.

3.5. Summary

To summarize, almost all injected redirect codes were developed in obfuscated JavaScript to conceal redirect destinations. Gradually increasing DNS lookup failures resulted from the large amount of domains that are obviously AGDs. The drive-by download sequences often involve an intermediate website to control redirections. To address **RQ1**, we describe the in-depth analysis of AGDs and controlling redirections in Section 4.

Domains related to drive-by downloads actively emerged on URL redirections in 2012 to 2013. While they were not so active in 2014 and later, the number of domains/URLs related to web advertising or tracking were increasing. This is the answer to **RQ2**, which indicates the change in the purpose of redirection based on redirect code injections, that is, click-fraud monetization is a new purpose in addition to malware injection.

4. In-depth analysis of evasive redirection techniques

To address **RQ1** regarding the key characteristics of the URL redirection mechanisms, we unveil and dissect evasive redirection techniques deployed by attackers in our observation.

The after-mentioned malicious techniques dedicated to the URL redirection ecosystem are deployed through collusion with each other. Our analysis revealed that attackers use the following techniques: *domain-flux* by *web-based* DGAs, *IP-flux* by fast-flux service networks (FFSNs), redirection controlling by traffic distribution networks (TDSs), and target profiling by tracking services. We give a clear overview of the URL redirection ecosystem revealed through our measurement in Fig. 7 and explain its characteristics as follows.

4.1. Web-based domain generation algorithm

The use of DGAs has become popular since 2008 due to the emergence of *Conficker* (Leder and Werner, 2009). Originally, DGAs were used for C&C in the *post-infectious* phase, for instance, malware-infected hosts communicate with specific generated domains, i.e., C&C server, for only a very short period of time. Although our analysis was focused on the *pre-infectious* phase, to our surprise, we discovered many suspicious AGDs in the observed redirect URLs. We inspect how DGA mechanisms have been used in the ecosystem of URL redirection. Our study revealed that DGAs are also used as a key component of the redirection mechanism. The use of the DGAs we observed for URL redirection is broadly classified into two categories: client-side domain generation (CDG) and server-side domain generation (SDG). We give details on CDG and SDG below.

4.1.1. Client-side domain generation

Web-based CDG is implemented as JavaScript. When a web browser accesses a page containing a JavaScript DGA, it runs on the browser and pseudo-randomly generates a domain name and its URL. After that, the JavaScript DGA outputs redirect code, e.g., an `iframe` tag set `src` attribute as a generated URL.

The JavaScript DGA is highly obfuscated to obstruct analysis. We used a browser emulator to fully execute obfuscated JavaScript and extract input/output values of `eval()` and `document.write()` as candidates of de-obfuscated *human-readable* code. We then manually analyzed this extracted

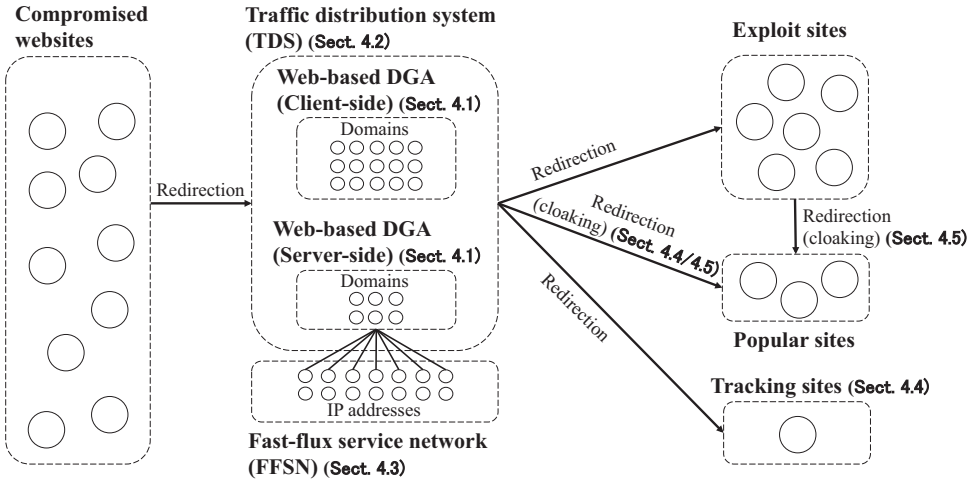

Fig. 7 – Observed malicious redirect ecosystem.

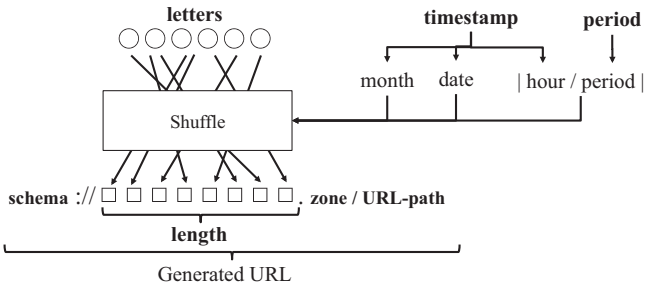

Fig. 8 – DGA: parameters and calculation.

JavaScript and identified the following parameters used for building redirect URLs including the pseudo-randomly generated domain names. We describe the parameters used in the JavaScript DGA as follows and show the procedure to generate a redirect URL in Fig. 8.

- **length**: This *length* determines the length of a randomly generated hostname.
- **letters**: These *letters* are candidates to be pseudo-randomly selected to construct a hostname.
- **timestamp**: The current *timestamp* is used as a seed of random function. It is not directly used, but *month*, *date*, and *hour* values extracted from the timestamp are used as input of the random function.
- **period**: A quotient of the *hour* and *period* is used for one of the inputs of a random function. This means that the outputs of the DGA changes every *period* (hours). For instance, if the *period* is set to 6, at most four ($24\ hours/6 = 4$) AGDs would be generated in a day.
- **zone**: A generated hostname is added as a prefix of strings of *zone*, which we call an "upper-level domain" throughout this work. For instance, given an AGD, `123abc.example.dga`, its upper-level domain is `example.dga`. Note that, in some cases, an upper-level domain of an AGD could be a top-level domain (TLD), such as `.ru` or `.pro`. Concatenated strings stand for a specific domain.
- **schema** and **URL-path**: *URL-path* is added to the above generated domain as its suffix. These strings mean the `<host>` and `<path>` parts of a URL. Finally, after *schema*, i.e., `http://` or `https://`, is added to these strings as its prefix, a specific URL is built.

The discovered instances were 16-character hostnames with `.info`, `.ru`, `.pro`, and .mynumber.org.

4.1.2. *Server-side domain generation*

In addition to web-based CDG, evidence of another type of web-based DGA, that is SDG, was found through our measurement. In the observed redirections, characteristic URL sets with random character hostnames with a fixed length and similar URL path (i.e., `count[1-9][0-9]?\.php`), which differs from the sets mentioned in Section 4.1.1, often emerged. However, only hard-coded URLs were set in the injected redirect code instead of a JavaScript DGA. We assume that an attacker previously executes a DGA to generate a domain name at the server-side and injects redirect code with the generated domain name into his or her own compromised website. The discovered instances were 7-character hostnames with `.ru` and `.com`, and 8-character ones with `.ru` and `.us`.

Table 7 – Comparison between C&C and web-based DGAs.

	DP	AM	CA	OC
C&C	Client (CDG)	Difficult	Available	Low
Web	Client (CDG)	Easy	Available	Low
	Server (SDG)	Easy	Unavailable	High

DP: deployment point; AM: algorithm modification; CA: code availability; OC: operation cost.

4.1.3. *DGA-type comparison and feasibility of conventional detection methods*

Countermeasures against DGAs, such as detecting and extracting AGDs and sinkholing, have already been implemented and contributed to disrupting C&C (Conficker Working Group, 2011; Shadowserver, 2014). We discuss to what extent the conventional countermeasures are effective for newly dissected DGAs. First, we organize the main difference between conventional DGAs and web-based DGAs and the difference between CDG and SDG. Fig. 9 gives an overview of C&C-based and web-based DGAs. A C&C-based DGA, which is conventionally used, is built in malware binary and used for opportunistically communicating with C&C servers. A web-based DGA is used for redirecting web users to malicious websites such as exploit sites.

The most significant difference is the deployment point, which means *where the DGA actually runs*. Table 7 shows a comparison between the types of DGAs in regards to algorithm modification, code availability, and operation cost. In a C&C-based DGA, an attacker implements a specific algorithm inside malware binary. After distributing malware, an attacker behind a C&C server waits to receive a callback from the malware-infected host. Therefore, an attacker cannot change the algorithms or parameters of the DGA before establishing C&C with the malware-infected host. In contrast, a web-based DGA is more flexible than a C&C-based DGA because an attacker can change the algorithm or its parameters on a compromised website whenever he or she wants.

It is generally known that DGAs use current time information, e.g., hour, day, and month, for the seed of random function. Since when and what the potential domain names are generated can be confirmed by an attacker, he or she previously registers potentially generated domains to operate domains working during specific times. A specific DGA is inside the client-side program that is malware binary, so that security researchers can analyze it by using a malware sandbox or reverse engineering and predict potential domain names that will be generated in the future (Conficker Working Group, 2011; Yadav et al., 2010). Therefore, a C&C-based DGA is potentially vulnerable to enumerating AGDs. Due to the same weakness web-based CDG has, we can enumerate AGDs serving web-based attacks when we obtain that DGA code.

Although it should be very costly for this SDG operation to frequently update redirect code, potential domain names of SDG are unpredictable, so security measures, such as domain takedown, are more difficult than those of CDG.

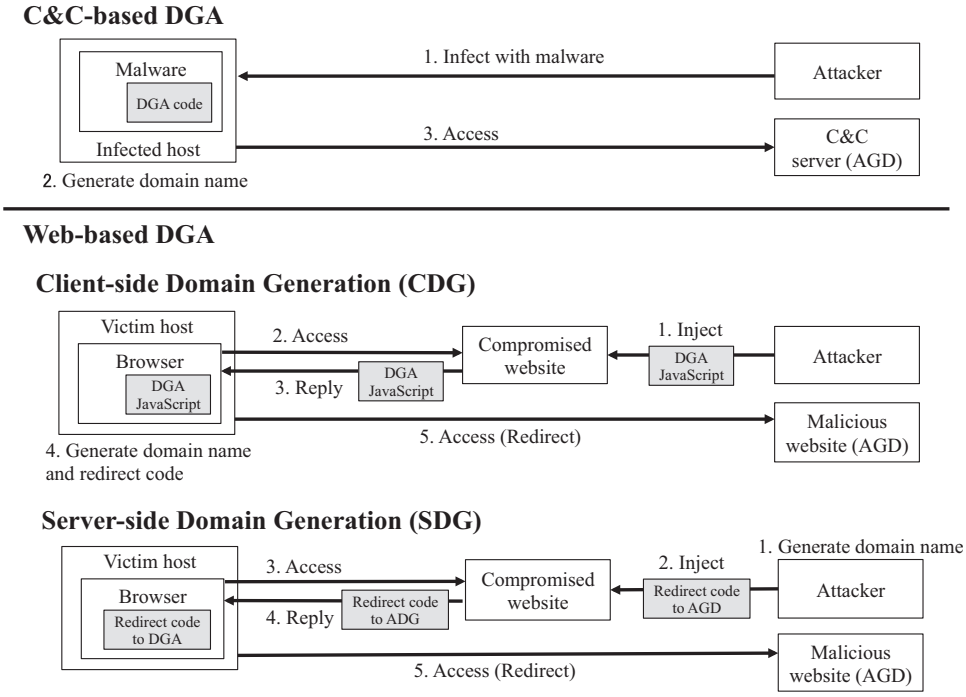

Fig. 9 – Overview of C&C-based and web-based DGAs.

In a C&C-based DGA, a large volume of DNS queries and failures generally occurs from each infected host as time progresses. Many detection methods have been proposed that are based on these characteristics (Antonakakis et al., 2012; Yadav and Reddy, 2011). However, these conventional methods may fail to detect web-based DGAs. In contrast to malware-infected host accessing C&C-based DGAs, a web user unexpectedly accesses web-based DGAs when he or she accesses compromised websites with injected redirect code. In web-based DGAs, accessing AGDs occurs when a web user occasionally accesses a landing site injected with redirect code. Therefore, the conventional viewpoints toward detection, such as using a large volume of NXDomain derived from speculative DNS queries and failover behavior, do not effectively work. While the volumes of DNS query and failure are not so large, the characteristic of domain name randomness still remains. Therefore, while conventional DGA detection methods based on linguistic features were implicitly targeting C&C-based DGAs, they may also be effective for web-based DGAs.

4.1.4. *Lifespan*

Generally, the more malicious websites are used for a long period, the more blacklisting is effective against them. To circumvent this blacklisting, an attacker accordingly changes redirect code on compromised websites to switch to redirect destinations. To understand the effectiveness of typical blacklisting, we estimated the lifespan of redirect destinations. The durations of CDG, SDG, and other *non-DGA* websites are separately shown in Fig. 10. We calculated domain A's lifespan $D(A) = T_f(A) - T_i(A)$, where $T_f(A)$ is the timestamp of the first

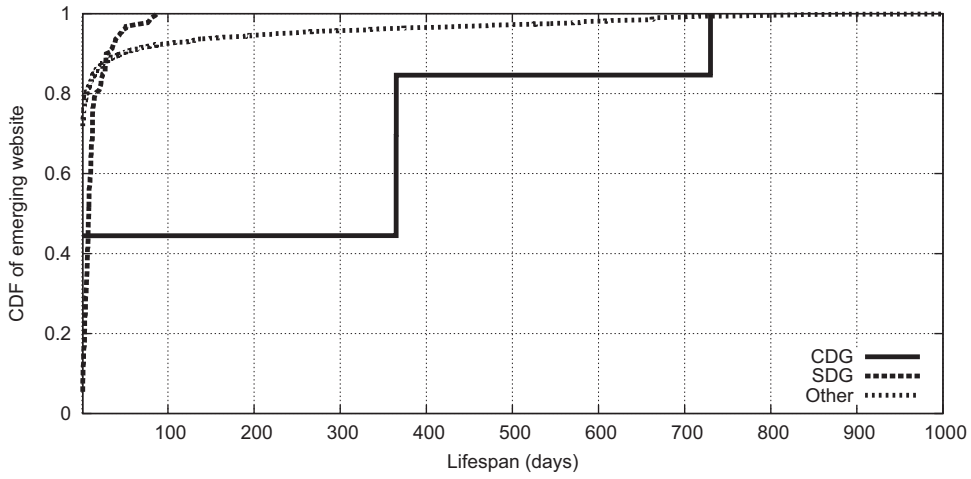

Fig. 10 – Lifespans of redirect websites.

emergence of A and $T_l(A)$ is the timestamp of the last emergence.

In the case of CDG, the two large gaps at 365 days (a year) and 730 days (two years) on the X-axis were caused by domains that emerge annually. The reason that annually emerging domains are also AGDs is explained in Section 4.1.1. The discovered instances of JavaScript DGA, i.e., specific JavaScript code, use only *month*, *date*, and *hour* information extracted from a current timestamp and do not use *year* information. Therefore, the same domain name is generated yearly on the same hour, date, and month, and we actually observed this phenomenon. The lifespans of SDG instances distributed a few to tens of days substantially different from those of CDG instances. Vanished websites on the redirections within 10 days were 82.6% out of other websites, while 7.5% were used over 100 days.

4.1.5. *DGA instances*

We heuristically classified AGDs on the basis of hostname length, upper-level domain, and URL-path similarity into several groups as DGA instances, which is a set of AGDs derived from a specific DGA. Table 8 shows summarized data corresponding to each instance. Instances **A1**–**A4** are of SDG, and all have a similar URL path. The remarkable point in these instances is that a massively large amount of IP addresses were resolved from their domain names. Instances **B1**–**B3** and **C** are of CDG, and all also have similar URL paths. Instances **B3** and **C** have a massively large amount of domain names, 1564 and 1919, respectively. Despite this, almost all domains of **B3** and **C** are unresolvable to IP addresses. Instance **C** has domains under a private suffix domain, so an attacker can freely create valid domains. In comparison, a domain registration to a registrar is required to validly use **B3**'s domain since **B3** is directly deployed under the TLD. Instances **B1**–**B3** had been used for only two or three weeks, then a resolvable domain never appeared. Instance **C** newly emerged after **B1**–**B3** operations. We manually checked to see whether **B1**–**B3** and **C** were deployed by the same DGA with the same parameters except for the upper-level domain. On the basis of this evidence, we assumed that the same attacker uses **B1**–**B3** and **C**. **B1** and **B2** were used in short periods. However, redirections using the JavaScript DGA of **B3** and **C** are actually discarded by an attacker and remain on compromised websites even if generated domains are unresolvable.

The domains and URLs of DGA instances account for only 33.8 and 3.4% of all redirect destinations, respectively; however, the domains of DGA instances account for 74.5% out of 2841 inspections detected as successful drive-by exploits. This means that web-based DGAs play an important role in drive-by exploits.

4.1.6. *Discarded redirect code*

In most cases, an attacker carefully maintains redirect code on a compromised website to change redirect destinations and obfuscation algorithms. However, we also discovered injected redirect codes and corresponding redirect destinations discarded by an attacker that have never been maintained or changed by the attacker from a certain time. If a web user accesses a website containing discarded redirect code, the redirection inevitably fails; therefore, it is actually harmless. In our observed objects, redirect codes toward 5 domains of **A1** and 25 domains of **A2** were discarded on several compromised websites, and these *ghost* redirections continued through each inspection. In addition, the JavaScript DGAs of **B1**–**B3** and **C** were also discarded and continued generating redirections toward unresolvable websites. Therefore, redirection failures continuously occur despite a discarded redirect code that has already been used by an attacker.

4.2. *Traffic distribution systems*

Our observed AGDs have two patterns of characteristic strings in the URL path: `count[1-9][0-9]?\.php` and `in.cgi\?[1-9][0-9]?`. One (`in.cgi`) is called *Sutra-TDS* in a previous paper (Symantec Security Response Blog, 2011), which is a toolkit for building a traffic distribution system (TDS).

A TDS is an intermediate website placed between an initial website (i.e., compromised website) and final destination website. The primary aim is to control the final redirect destinations to obscure them. In the observed data, we confirmed that final websites are frequently changed by a TDS as time progresses.

All our discovered DGA instances performed as TDSs and mediated between an initial website and an exploit site. Almost all redirect methods on the AGDs based on **A1**–**A4** were HTTP-200 with JavaScript location.href or HTTP-302.

4.3. *Fast-flux service network under AGDs*

Domains generated by SDG (**A1**–**A4**) respond with different IP addresses in each DNS resolution. Consequently, they had an extremely large amount of IP addresses through our

Table 8 – Discovered DGA instances.

	First seen	Last resolvable	Last seen	UD	HL	URL-path	DGA type	# of domains (IP resolvable)	# of URLs	# of IPs	Prevalence (%)
A1	'12-04-10	'12-08-01	Ongoing	.ru	7	count**.php	Server	36 (36)	36	1,275	3.9
A2	'12-12-08	'13-08-05	Ongoing	.ru	8	count**.php	Server	131 (128)	131	35,789	69.8
A3	'13-05-20	'13-06-21	'13-06-21	.us	8	count**.php	Server	39 (39)	49	1,510	54.2
A4	'13-05-31	'13-06-17	'13-06-20	.com	7	count**.php	Server	16 (16)	16	2,111	54.8
B1	'12-07-11	'12-09-28	'12-10-03	.info	16	in.cgi?**	Client	43 (23)	43	8	1.9
B2	'12-08-02	'12-08-27	'12-09-12	.ru	16	in.cgi?**	Client	53 (32)	53	7	1.9
B3	'12-10-03	'12-10-20	Ongoing	.pro	16	in.cgi?**	Client	1,564 (16)	1,564	2	1.9
C	'12-11-08	'14-12-05	Ongoing	.mynumber.org	16	in.cgi?**	Client	1,919 (11)	1,919	5	1.4

UD: upper-level domain; HL: hostname length.

measurement, as mentioned in Section 4.1.5 and shown in Table 8. We observed over 1K IP addresses for each instance and 39,131 in total. The following evidence indicates that **A1**–**A4** are deployed on a *fast-flux service network* (FFSN). As seen from the `GeoIP` information, these IP addresses are massively globally distributed, for example, **A1**–**A4** include 425 autonomous system numbers (ASNs) in 95 countries, 1849 ASNs in 117 countries, 374 ASNs in 48 countries, and 414 ASNs in 56 countries. We calculated the ASNs of all IP addresses on FFSNs and manually checked the several top ASNs and their usages. Most IP addresses are located on residential networks so that end user PCs are the most dominant. While the IP addresses are globally distributed on hundreds of ASes, Ukraine's telecom/mobile networks (AS15895 and AS25229) account for over 22%. The report published by RiskAnalytics (2016) mentioned that their originally observed IP addresses of an FFSN are also located at the same ASes we observed so that we recognized the commonly used bot-infected PCs for FFSNs. The emerging periods of these instances are temporally diversified; however, 1557 IP addresses out of all resolved IP addresses are multiply used by them.

Hosts of resolved IP addresses successfully respond with HTTP replies in most cases; in other words, they have valid IP addresses and run as a website. The percentages of IP addresses to which HTTP successfully responded in **A1**–**A4** are 79.5, 91.0, 92.6, and 80.6%, respectively. Therefore, in most cases, the authoritative DNS servers of FFSN domains in **A1**–**A4** faithfully answer valid IP addresses controlled by attackers.

All reply headers of HTTP include the following four commonly used specific header fields: `Server:Apache`, `Content-Type`: without a specific content type, `Server:nginx/1.2.6`, and `X-Powered-By:PHP/5.4.11`. These reply headers are slightly weird because `Server:` fields are multiply defined, and the `Content-Type:` field is always null. In consideration of this evidence, FFSN agents seem to be commonly installed on a specific server or use *blind proxy redirection* (BPR). It is said that BPR is typically used for an FFSN (Honeynet Project, 2007), which works as a reverse proxy to transparently send received HTTP requests to a backend server directly controlled by an attacker. In this way, an attacker directly knows accessed hosts' information and collectively switches HTTP replies at a single point without exposure.

4.4. *Double-crop redirecting*

One notable point is that *Google Analytics* emerged on redirect chains derived from about 60% of compromised websites. Google Analytics is one of the most well-known web access analysis services. The emerging Google Analytics are from advertising pages and drive-by related pages. The former is normal usage that aims to collect web users' profiles to prepare to deploy more effective personalized ads. The latter's aims seems to be profiling targeted web users and further strategize effective exploits or monetization. In our content analysis, the web content of 19 domains in **A1** contained Google Analytics' JavaScript and redirected web users to www.google-analytics.com. These 19 domains were set as redirect destinations on 43.4% of compromised websites. In addition, the same tracking ID, obviously an attacker's, was commonly used in all these domains. This tracking ID is embedded as a plain text in web content; therefore, we can easily identify it.

4.5. *Thwarting security inspection*

One of the reasons for HTTP access failure seems to be *cloaking*. Although cloaking was originally used for SEO poisoning, it is currently used for web-based malware infection. To thwart security inspection, malicious websites respond with harmless content or redirect to a legitimate website if they recognized an access as a security inspection. The most popular cloaking technique is IP cloaking, which identifies client IP addresses, e.g., detected repeated accesses or blacklisted IP addresses, and flexibly responds (Rajab et al., 2011). It has been reported that almost all exploit kits have IP cloaking functionalities (Eshete and Venkatakrishnan, 2014). In addition, a security vendor's report mentioned that TDSs also conduct cloaking (TrendMicro, 2011). To bypass IP cloaking, we make an effort to obfuscate the IP addresses of a honeyclient by using web proxies distributed on various ASes. *User-agent* information and referrers are also usually used for cloaking. Our honeyclient is a high interaction system, so the user-agent information and referrer are naturally set by an actual web browser. Therefore, we did not make any special effort to obfuscate this information. Note that our monitoring system did not access websites through a different browser so that it also possibly fails to access websites which change redirect destinations based on user-agent information.

In our inspection, our honeyclient encountered several types of cloaking. We summarize probable cloaking situations in HTTP as follows.

- HTTP error: HTTP-404 or HTTP-500 responses often occur in inspections of exploit sites. These types of response are typical cloaking behavior of exploit kits.
- HTTP-200 without meaningful content
  Malicious websites respond with HTTP-200 without meaningful content, for instance, 0-byte content or OK. When a web browser receives such a reply, no redirection or exploit occurs. More than half of the HTTP responses from **A1**–**A4** were HTTP-200 with 0-byte content.
- HTTP-302 redirect to benign websites
  Malicious websites respond with HTTP-302 with specific redirect URLs, which are Google sites, Microsoft sites, or *localhost*. In addition, on some exploit sites, both exploits and HTTP-302 redirects whose redirect destinations are popular websites simultaneously occurred. This seems to intentionally confuse security inspection. In many cases of redirecting to benign websites, the top pages of popular websites, such as http://www.google.com/, http://www.google.se/, and http://www.bing.com/, are used for HTTP-302 redirect destinations.

Some redirections of HTTP-302 were not sophisticated because they were falsely set to redirect destinations as http://google.com/ or http://bing.com/, which are not actually accessible. When such *non-existent* URLs are falsely accessed by a web user, the URL redirects the user to the top effective URLs such as http://www.google.com/ or http://www.bing.com/.

4.6. Summary

To summarize, our in-depth analysis unveiled evasive techniques to ensure significantly tolerant operation of URL redirections: domain-flux, IP-flux, redirection controlling, and target profiling. Domain-flux by web-based DGAs generated over three thousand AGDs over an entire period. We found two uses of DGAs: CDG and SDG, and discussed the operational difference between the types of DGAs and effectiveness of conventional methods. These observed AGDs performed as TDSs which are intermediate websites placed between an initial website and a final destination website to control final redirect destinations and obscure them. IP-flux by an FFSN and the above domain-flux (especially CDG) are concurrently used for deploying TDS. In addition, to profile victim web users and to further strategize effective exploits or monetization, intermediate websites (i.e., TDSs) use a web tacking service while redirecting to exploit sites. The characteristics of these techniques to ensure tolerant operation of URL redirections are the answers to **RQ1**.

5. Mitigation

We introduce simple and practical countermeasures against URL redirect injection based on our knowledge obtained from our large-scale and long-term observation.

5.1. Countering DGAs

For countering CDG, AGDs generated in the future can be predictable, once we acquire the DGA code. Therefore, information obtained from our honeypot-based monitoring system can be used for existing blacklisting and domain takeover operation. In contrast, the code of SDG is not available because it is previously executed in an attacker-side environment.

Many methods for detecting AGDs require training data that contain a set of actual malicious domain names. The conventional method to acquire training data is analyzing a malware binary to extract AGDs generated by a C&C-based DGA. Similarly, our measurement system can acquire a substantial volume of AGDs by analyzing the web-content injected code of web-based DGAs.

5.2. Discovering unknown compromised websites via sinkholed HTTP requests

A domain takeover can observe the accesses from malware-infected hosts to a sinkholed domain of a C&C server and harvest theft sensitive information (Stone-Gross et al., 2009). An HTTP request is generally an attached origin URL used as a referrer field; therefore, we infer that a domain takeover against malicious redirect destination websites can collect victims' HTTP requests and discover unknown origin websites that are compromised websites.

The HTTP request header of redirection is attached with referrer information, which is the origin of the redirection, i.e., the URL of a compromised website injected with redirect code. We confirmed that all HTTP requests toward websites of DGA instances are attached with correct referrer fields. If we previously know the URL of a specific malicious website, it is possible to discover unknown compromised websites by checking the referrer information contained in an HTTP request toward the malicious website. Conceivable method of monitoring such a request is using web proxy logs or website takeovers by domain sinkholing. We can enumerate potential domains used in the future by reversing client-side DGAs. Given that almost all domains are not registered and are unused, we can legitimately register potential domains to own a part of the AGDs instead of an attacker. If we successfully complete domain registration, on our website for observation, we can receive HTTP requests from potential victim web users when our DNS server responds with the A record set as an IP address of our website.

The phenomenon that discarded redirect code permanently remains on compromised websites is mentioned in Section 4.1.6. Although discarded redirect code is not currently harmful, there is a potential risk that valid redirect code will be injected some day. Therefore, compromised websites should be fixed immediately after, even if redirect code seems to be discarded.

5.3. Disabling attackers' advertising and tracking IDs

Redirections for web ads and tracking are not inherently malicious, but all redirections included on compromised websites are obviously implied by an attacker's malicious intention. Injected redirect code for web advertising and tracking sometimes contains IDs, e.g., Google Analytics' tracking ID represented as `UA-000000-01`. We found some IDs in injected codes across different compromised websites, and these IDs obviously belong to certain attackers. One action we can take is reporting such attackers' IDs and the evidence of compromising, i.e., fraudulent access log and injected code, as violating terms of use to advertising or tracking service providers to ban the IDs.

5.4. Suspicious redirect path detection based on cloaking behavior

We assume that accessing the top pages of prominent websites by HTTP-302 redirect is not a common occurrence and this notion supports the focusing of attention on suspicious redirect by cloaking behavior. We surveyed benign redirections from popular websites with our honeyclient. It inspected Alexa 607K websites' top pages with corresponding URLs (22M URL accesses, 15M unique URLs). In these inspections, HTTP-3xx redirects occurred 693K times. Regarding www.google.com, although there are many redirects pointing to variable URL-paths under the domain, only 11 redirects (origin URLs) point to the top page. The reason of the above redirections in 6 out of the 11 origin URLs is closed or moved websites. We conclude that a redirect pointing to a top page of a prominent website is extremely rare. We recommend that HTTP-3xx redirects to the top pages of prominent websites should be carefully investigated.

5.5. *Producing dataset for security research and operation*

We statistically analyzed observed data to give an overview of the ecosystem of malicious URL redirection and changes throughout a prolonged period. Countering in real time is out of the scope of this study; however, it can contribute to generating a dataset for supporting the development/evaluation of detection tools and security operations. The MALICIA project[4] (MALICIA Project, 2013; Nappa et al., 2015) and malware-traffic-analysis.net (http://www.malware-traffic-analysis.net/) are known to provide datasets of web-based attacks. The MALICIA dataset has been provided to many research organizations. malware-traffic-analysis.net also presents training excises to analyze malicious content for security researchers/engineers.

Related work, e.g., crawling benign websites (Lee and Kim, 2012; Li et al., 2012) or monitoring merged HTTP traffic of end users (Stringhini et al., 2013), had the difficulty of finding malicious entities from a large amount of data that mainly contain benign entities. In contrast, our observed data originated from a malicious activity (i.e., redirect injection) and contain less noise (i.e., benign redirection) than those of the above studies.

6. Discussion

6.1. *Generality of observed data*

We believe that collected malware executables for malware analysis and induced redirect injection have generality for the following reason. When we leak honeytokens to various attackers, we repeatedly collected initial malware executables from various blacklisted URLs, which are included in the public blacklist (malwaredomainlist.com) widely used by security engineers, and a huge number of general public websites.

In addition, to examine how our observation results reflect the real world, we surveyed how many redirect websites actually emerged on compromised websites in the wild. We used another source by crawling data related to the compromised websites that were originally benign. The data were obtained from our honeyclient that inspected the top pages of about 160,000 websites in two- or three-day intervals between Aug. 2011 and Mar. 2015. It discovered 126 compromised websites performing as landing sites of drive-by downloads and containing 2512 websites (16,038 URLs) as redirect destinations. To concentrate on important malicious websites, we focused on DGA instances. We extracted redirect URLs that were matched with the patterns of DGA instances on the basis of a specific combination of hostname length, upper-level domain, and URL-path similarity previously indicated in Table 8. The numbers of emerged websites for each DGA instance are listed in Table 9. To understand the overlap between them, in addition to the numbers of simply overlapped AGDs, we calculated two types of similarity; Simpson index, $Sim_S(X, Y)$ and Jaccard index, $Sim_J(X, Y)$, which are computed as follows:

[4] This project has stopped distributing a dataset due to the aging of the dataset and the students in this project graduating.

Table 9 – Overlap with compromised legitimate websites.

	# emerged in decoy site	# emerged in legitimate sites	# of overlaps	Sim_S	Sim_J
A1	36	34	28	0.88	0.83
A2	131	85	67	0.78	0.51
A3	39	4	4	1.00	0.10
A4	16	11	7	0.63	0.43
B1	43	41	6	0.14	0.13
B2	53	47	6	0.12	0.11
B3	1564	24	13	0.54	0.00
C	1919	90	49	0.53	0.04

$$Sim_S(\mathbf{X}, \mathbf{Y}) = \frac{|\mathbf{X} \cap \mathbf{Y}|}{|\min(\mathbf{X}, \mathbf{Y})|}, \quad Sim_J(\mathbf{X}, \mathbf{Y}) = \frac{|\mathbf{X} \cap \mathbf{Y}|}{|\mathbf{X} \cup \mathbf{Y}|}$$

These indices are usually used to measure the similarity of sets.

As the AGDs of all DGA instances emerged on redirect-injected compromised websites, there was a little gap between our observed URL redirect injections and those of the real world. In other words, this does not mean that our observation was focused on narrow and not peculiar space. In fact, there were various AGDs discovered with only our system. We assumed that decoy-based observation effectively works to approximately understand overall circumstances. Although it is easy to discover compromised legitimate websites by using decoy websites, we can observe more AGDs than those of compromised legitimate websites. The observation stability of our decoy websites contributes to such broad coverage.

6.2. *Impact on real world*

We surveyed how many times discovered redirect websites were actually accessed by the general public users to examine the impact on the real world. To conduct this survey, we used the DNSDB (DNSDB) gathering passive DNS logs collected from globally distributed distinct organizations. The passive DNS logs included a DNS query toward domains from users under specific cache DNS servers. Therefore, we could determine how many users actually accessed a specific domain under a specific cache DNS server by counting the various replies with the A records. Impact estimation with passive DNS logs has been extensively investigated (Grier et al., 2012; Schiavoni et al., 2014). Similarly, we conducted the aforementioned survey in July 2015. A user of DNSDB can retrieve a summarized resource record, resource data, and the access number with timestamps, which are the first seen and last seen ones. We counted the number of accesses in the `A record` except for unresolvable DNS queries. We retrieved the number of DNS queries toward specific domains from the DNSDB, which is a database indexing DNS queries collected from cache DNS servers and authoritative DNS servers used by various organizations.

Even though a user who receives an `A record` of a specific domain does not always access that domain, we simply counted the number of valid queries in the DNS layer. Table 10 shows the number of successful DNS queries (authoritative DNS server replying with an `A record`) in each DGA instance and

Table 10 – Query to AGD in DNSDB.

	# of domains (IP resolvable)	# accessed in DNSDB	# accessed IPs in DNSDB	Sim_S of IP	Sim_J of IP
A1	36 (36)	90,507	28,354	0.57	0.02
A2	131 (128)	147,855	34,502	0.35	0.34
A3	39 (39)	8,748	2,346	0.44	0.28
A4	16 (16)	4,926	1,697	0.40	0.32
B1	43 (23)	8,759	19	0.87	0.36
B2	53 (32)	1,887	9	1.00	0.77
B3	1,564 (16)	1,024	2	1.00	1.00
C	1,919 (11)	241	5	1.00	1.00

Table 11 – Top 10 high impact domains of exploit sites.

Domain	Role	# accessed in DNSDB
warpdriveactive[dot]com	Exploit site	1,109,311
stevebeam[dot]com	Exploit site	333,166
bluefuse[dot]com	Exploit site	265,228
prepaidphoneguy[dot]com	Exploit site	142,540
rompnroll[dot]com	Exploit site	127,733
vistaclues[dot]com	Exploit site	114,418
lovedbaby[dot]com	Exploit site	109,606
jimmyhophotography[dot]com	Exploit site	108,620
chelmsfordlibrary[dot]org	Exploit site	104,727
laurendavidstyle[dot]com	Redirector	102,630

overlap comparisons between each set of IP addresses. Instances **A1–A4** were surprisingly accessed about 252K times, **B1–B3** were accessed about 11K times, and **C** was accessed only 241 times. The same sets of IP addresses in **B1–B3** and **C** appeared in the DNSDB.

A large amount of IP addresses of **A1–A4** were also resolved in the DNSDB. Although overlaps indicate high similarity with the set of IP addresses observed in the decoy, there were unique entities, i.e., IP addresses, observed using distinct methods. We had further interest in the question, "*How many IP addresses (agents) does this FFSN actually use?*". To answer this question, we used *mark and recapture* estimation, which is a method for estimating a population's size and is commonly used in ecology. The *Lincoln–Petersen estimator* (Schwarz and Seber, 1982) represents $\hat{N} = Kn/k$, where $\hat{N}$ is the estimator of AGDs in a population, K is the observed number of entities captured using the 1st method, i.e., decoy in this case, n is the observed number of entities captured using the 2nd method, i.e., the DNSDB in this case, and k is the observed number of recaptured entities that were marked. By substituting the observed number of IP addresses, i.e., $K = 39{,}131$, $n = 65{,}356$, and $k = 14{,}022$, to the estimator, we get the estimation as $\hat{N} \simeq 182\,\mathrm{K}$ IP addresses, which is the estimated population of our observed FFSN agents. The estimated population is much larger than that of known FFSNs discovered in previous surveys (Holz et al., 2008; Passerini et al., 2008).

We additionally surveyed the high impact websites involved in the exploit-detected inspections shown in Table 11. We manually analyzed inspection logs and found that all these websites were actually attributed to either exploit sites or redirectors. All these domains, except for `warpdriveactive` and `laurendavidstyle`, were accessed via the TDS of **A2**.

6.3. *Attack automation*

Attackers try to automate each malicious activity to conduct a sequence of attack timely and scalably. Several types of code to steal credentials are publicly available. Metasploit provides modules to steal credentials from various client applications (https://github.com/rapid7/metasploit-framework/tree/master/modules/post/windows/gather/credentials). One malware program called Pony, also known as Fareit, has the functionality of information stealing (Trustwave, 2013), and its source code was leaked and is now publicly available (https://github.com/nyx0/Pony). Due to the availability of credential-stealing code, various types of malware can be easily implemented with such functionality. In the injection activity, an attacker first downloads web content from a Web CMS then uploads modified web content with redirect code to it. Attackers probably use certain automatic tools instead of manual operation because much of this execution time of code injection is extremely short, for example about 80% of execution time of code injection is less than 2 seconds. The websites that are redirect destinations obviously have exploit-kit-derived URLs. In our previous study, we manually analyzed the data in the first year and confirmed five popular exploit-kits at that time, i.e., Blackhole, Redkit, Phoenix, Incognito, and Neosploit, used on redirect destinations (Akiyama et al., 2013). Our monitoring system does not depend on specific detection signatures, so it has the potential for collecting the latest malicious entities regardless of the type of toolkit.

6.4. *Adaptivity*

Our monitoring system can be applicable to various types of applications/services as long as they require ID/password authentication. Secure FTP (SFTP) and secure shell (SSH) are possible expansions of our monitoring system. We should change only two settings to apply them: preparing corresponding honeytokens (e.g., putting a configuration file of SFTP or SSH on a malware sandbox), and running the services on a honeypot awaiting fraudulent login. However, to apply our monitoring system to public online services (e.g., social networks, online banking, online shopping, Webmail), we must cooperate with each service provider to monitor behind the service.

7. Related work

7.1. *URL redirection analysis*

Many studies have been conducted to examine, detect, and discover URL redirects on specific services, the server-side, client-side, and honeypot.

WarningBird reveals that redirect networks start from Twitter URLs (Lee and Kim, 2012). MadTracer inspects the top 90K popular websites for several months and reveals the ad network structure and characteristics (Li et al., 2012). These monitoring systems start from benign websites; therefore, they require large-scale crawling. In contrast, our monitoring system starts from decoy websites with injected malicious redirect code so

that it can consistently track each URL redirection without large-scale crawling.

SpiderWeb (Stringhini et al., 2013) analyzes the web access data of actual web users on client hosts and detects malicious websites on the basis of different web users' redirect structures. This system requires web accesses from various client environments and uses the web access data of anti-virus programs installed on computers that are not publicly available and affected by privacy issues. Our data are obtained from our managed decoy servers and are not affected by privacy issues.

While our monitoring system focuses on server-side web injection, Hulk is an analysis system that detects web injection on web browsers (not websites), which is caused by malicious browser extensions (Kapravelos et al., 2014). Thomas et al. focused on the aforementioned client-side web injection and conducted large-scale observation (Thomas et al., 2015).

Canali et al. developed decoy websites with vulnerable web applications to incur intrusions via vulnerabilities and surveyed how intruders compromise websites (Canali and Balzarotti, 2013). Akiyama et al. developed a decoy system that prompts malware to exfiltrate bait credentials and lure attackers into decoy websites with stolen credentials to effectively collect compromised web content (Akiyama et al., 2013). However, understanding the ecosystem of URL redirects on compromised websites was out of the scope of these studies.

7.2. *DGA detection and analysis*

Malicious domain names are often generated using DGAs to build a resilient infrastructure for conducting malicious activities. Many researchers have reported that various notorious malware families, such as Kraken/Bobax, Conficker, Murofet, Mebroot/Torpig, Srizbi, Bonnana, and Zeus, usually use DGAs for generating their C&C servers' domain names (Antonakakis et al., 2012; Bilge et al., 2011; Damballa, 2012; Schiavoni et al., 2014).

Pseudo-randomly generated domain names obviously have a specific linguistic feature that differs from that of *human-generated* domain names. Many researchers have proposed DGA-detection methods based on the linguistic features of domain names (Schiavoni et al., 2014; Yadav et al., 2012). In contrast, detection methods based on DNS traffic features, such as a large volume of NXDomains, have also been proposed (Antonakakis et al., 2012; Yadav and Reddy, 2011).

The weakness of a conventional DGA is that potential domain names generated in the future will be be easily exposed if the algorithm is analyzed. Many botnet takedown operations have been conducted using domain sinkholing based on extracted potential domains. In addition, Stone-Gross et al. observed botnet communication toward C&C by using sinkholed domains and collected information about bot-infected hosts and stolen data (Stone-Gross et al., 2009).

Research on DGAs focusing on a botnet's C&C, i.e., post-infectious phase, has been extensive; however, little DGA research has been focused on the pre-infectious phase. Our study is the first to conduct longitudinal observation in the pre-infectious phase. Through our observation, we reveal the actual condition of using web-based DGAs and organize the main differences between conventional DGAs and web-based DGAs in Section 4.1.3.

7.3. *Secure web-application framework against credential theft*

Strong authentication, such as two-factor authentication (TFA), is effective for protecting a system from a fraudulent login using stolen credentials. Drupal (Drupal, 2011), a well-known CMS, has the functionality of TFA, so a server using this CMS can provide secure authentication instead of simple ID/password authentication such as FTP.

8. Conclusion

In this work, we attempted to shed light on the ecosystem of malicious redirections started from compromised websites. In particular, we focused our attention on the core mechanism of web-based attacks – URL redirection. We investigated the following research questions: **RQ1**: *What are the key characteristics of URL redirection mechanisms?*, and **RQ2**: *Have their purposes been changed over time?* The main contribution of this work is the deployment of a honeypot-based monitoring system to track the ecosystem of URL redirections for a long period. Through the extensive analysis of longitudinal data collected with our monitoring system across four years, we derived the following findings. The findings corresponding to **RQ1** are (A) the URL redirection mechanism exhibited intrinsic change and new trends, (B) the use of web-based DGAs has become popular as a means to increase the entropy of redirect URLs, and (C) both domain-flux and IP-flux are concurrently used for deploying the intermediate sites of redirect chains to ensure robustness of redirection. The findings corresponding to **RQ2** are (D) click-fraud monetizing has recently become a new purpose of attacker in addition to malware infection, and (E) interestingly, we found that web tracking services that track the statistics of visitors, i.e., victims, are installed onto redirect URLs. To evaluate the generality of our observation, we quantified the impact of the malicious URL redirection mechanism in the real world by correlating locally and globally collected data.

On the basis of these findings originating from the unveiled URL redirection ecosystem, we also presented practical countermeasures against malicious URL redirections. Security/network operators can leverage information obtained from the honeypot-based measurement method to conventional security operations: disrupting infrastructures of web-based attack by using domain blacklisting/takedown, report web advertising/tracking IDs, and discovering victims such as unknown compromised websites in the web by using domain takeover.

REFERENCES

Akiyama M, Aoki K, Kawakoya Y, Iwamura M, Itoh M. Design and implementation of high interaction client honeypot for drive-by-download attacks. IEICE Trans Commun 2010;E93-B:1131–9.

Akiyama M, Yagi T, Aoki K, Hariu T, Kadobayashi Y. Active credential leakage for observing web-based attack cycle. In: Proceedings of the 16th international symposium on research in attacks, intrusions, and defenses (RAID); 2013.

Antonakakis M, Perdisci R, Nadji Y, Vasiloglou N, Abu-Nimeh S, Lee W, et al. From throw-away traffic to bots: detecting the rise of DGA-based malware. In: Proceedings of the 21st USENIX security symposium (Security); 2012.
Araujo F, Hamlen KW, Biedermann S, Katzenbeisser S. From patches to honey-patches: lightweight attacker misdirection, deception, and disinformation. In: Proceedings of the 20th ACM conference on computer and communication security (CCS); 2014.
Bilge L, Kirda E, Kruegel C, Balduzzi M. EXPOSURE: finding malicious domains using passive DNS analysis. In: Proceedings of the 2011 network and distributed system security symposium (NDSS); 2011.
Blizard T, Livic N. Click-fraud monetizing malware: a survey and case study. In: Proceedings of the 7th international conference on malicious and unwanted software (MALWARE); 2013.
Canali D, Balzarotti D. Behind the scenes of online attacks: an analysis of exploitation behaviors on the web. In: Proceedings of the 2013 network and distributed system security symposium (NDSS); 2013.
Conficker Working Group. Lessons learned June 2010. Published January. 2011. Available from: http://www.confickerworkinggroup.org/wiki/uploads/Conficker_Working_Group_Lessons_Learned_17_June_2010_final.pdf.
Damballa. DGAs in the hands of cyber-criminals; 2012. Available from: https://www.damballa.com/downloads/r_pubs/WP_DGAs-in-the-Hands-of-Cyber-Criminals.pdf.
DNSDB. Farsight security. Available from: https://www.dnsdb.info.
Drupal. Two-factor authentication (TFA); 2011. Available from: https://www.drupal.org/project/tfa.
Durumeric Z, Kasten J, Adrian D, Halderman JA, Bailey M, Li F, et al. The matter of heartbleed. In: Proceedings of the 2014 conference on internet measurement conference (IMC); 2014.
Eshete B, Venkatakrishnan VN. WebWinnow: leveraging exploit kit workflows to detect malicious URLs. In: Proceedings of the 4th ACM conference on data and application security and privacy (CODASPY); 2014.
fb1h2s. Sandy: opensource exploit analysis framework; 2014. Available from: https://github.com/fb1h2s/sandy.
Grier C, Ballard L, Caballero J, Chachra N, Dietrich CJ, Levchenko K, et al. Manufacturing compromise: the emergence of exploit-as-a-service. In: Proceedings of the 19th ACM conference on computer and communication security (CCS); 2012.
Holz T, Gorecki C, Rieck K, Freiling FC. Measuring and detecting fast-flux service networks. In: Proceedings of the 2008 network and distributed system security symposium (NDSS); 2008.
Honeynet Project. 2008. Capture-HPC.
Honeynet Project. Know your enemy: fast-flux service networks; 2007. Available from: http://www.honeynet.org/papers/ff.
Invernizzi L, Benvenuti S, Cova M, Comparetti PM, Kruegel C, Vigna G. EvilSeed: a guided approach to finding malicious web pages. In: Proceedings of the 2012 IEEE symposium on security and privacy (SP); 2012.
Kapravelos A, Grier C, Chachra N, Kruegel C, Vigna G, Paxson V. Hulk: eliciting malicious behavior in browser extensions. In: Proceedings of the 23rd USENIX security symposium (Security); 2014.
Leder F, Werner T. Know your enemy: containing conficker; 2009.
Lee S, Kim J. WarningBird: detecting suspicious URLs in Twitter stream. In: Proceedings of the 2012 network and distributed system security symposium (NDSS); 2012.
Li Z, Zhang K, Xie Y, Yu F, Wang X. Knowing your enemy: understanding and detecting malicious web advertising. In: Proceedings of the 2012 ACM SIGSAC conference on computer and communications security (CCS); 2012.
MALICIA Project; 2013. Available from: http://malicia-project.com.
Moshchuk A, Bragin T, Gribble SD, Levy HM. A crawler-based study of spyware on the web. In: Proceedings of the 2006 network and distributed system security symposium (NDSS); 2006.
Nappa A, Rafique MZ, Caballero J. The MALICIA dataset: identification and analysis of drive-by download operations. Int J Inf Secur 2015;14(1):15–33.
Passerini E, Paleari R, Martignoni L, Bruschi D. FluXOR: detecting and monitoring fast-flux service networks. In: Proceedings of the 5th international conference on detection of intrusions and malware, and vulnerability assessment (DIMVA); 2008.
Provos N, Mavrommatis P, Rajab MA, Monrose F. All your iFRAMEs point to us. In: Proceedings of the 17th conference on security symposium (Security); 2008.
Rajab MA, Ballard L, Jagpal N, Mavrommatis P, Nojiri D, Provos N, et al. Trends in circumventing web-malware detection; 2011. Available from: http://static.googleusercontent.com/media/research.google.com/ja//archive/papers/rajab-2011a.pdf.
RiskAnalytics. Dark cloud network facilitates crimeware; 2016 Available from: https://www.riskanalytics.com/blog/post.php?s=2016-08-17-dark-cloud-network-facilitates-crimeware.
Schiavoni S, Maggi F, Cavallaro L, Zanero S. Phoenix: DGA-based botnet tracking and intelligence. In: Proceedings of the 11th international conference on detection of intrusions and malware, and vulnerability assessment (DIMVA); 2014.
Schwarz CJ, Seber GAF. The estimation of animal abundance and related parameters; 1982.
Shadowserver. Gameover Zeus; 2014. Available from: https://goz.shadowserver.org/.
Spitzner L. Honeytokens: the other honeypot, 2003. Available from: http://www.symantec.com/connect/articles/honeytokens-other-honeypot.
Stone-Gross B, Cova M, Cavallaro L, Gilbert B, Szydlowski M, Kemmerer R, et al. Your botnet is my botnet: analysis of a botnet takeover. In: Proceedings of the 16th ACM conference on computer and communications security (CCS); 2009.
Stringhini G, Kruegel C, Vigna G. Shady paths: leveraging surfing crowds to detect malicious web pages. In: Proceedings of the 2013 ACM SIGSAC conference on computer and communications security (CCS); 2013.
Symantec Security Response Blog. Web-based malware distribution channels: a look at traffic redistribution systems, 2011. Available from: http://www.symantec.com/connect/blogs/web-based-malware-distribution-channels-look-traffic-redistribution-systems.
Thomas K, Bursztein E, Grier C, Ho G, Jagpal N, Kapravelos A, et al. Ad injection at scale: assessing deceptive advertisement modifications. In: Proceedings of the IEEE symposium on security and privacy (SP); 2015.
TrendMicro. Traffic direction systems as malware distribution tools, 2011. Available from: http://www.trendmicro.com/cloud-content/us/pdfs/security-intelligence/reports/rpt_malware-distribution-tools.pdf.
Trustwave. Look what I found: Moar Pony! 2013. Available from: https://www.trustwave.com/Resources/SpiderLabs-Blog/Look-What-I-Found—Moar-Pony!/.
Websense Security Labs. Mass injection – Nine-Ball compromises more than 40,000 Legitimate Web sites, 2009.
Yadav S, Reddy ALN. Winning with DNS failures: strategies for faster botnet detection. In: Proceedings of the 7th international ICST conference on security and privacy in communication networks (SecureComm); 2011.
Yadav S, Reddy AKK, Reddy ALN, Ranjan S. Detecting algorithmically generated malicious domain names. In: Proceedings of the 2010 conference on internet measurement conference (IMC); 2010.

Yadav S, Reddy AKK, Reddy ALN, Ranjan S. Detecting algorithmically generated domain-flux attacks with DNS traffic analysis. IEEE/ACM Trans Netw 2012;20(5):1663–77. http://dx.doi.org/10.1109/TNET.2012.2184552.
Zhang J, Yang C, Xu Z, Gu G. PoisonAmplifier: a guided approach of discovering compromised websites through reversing search poisoning attacks. In: Proceedings of the 15th international symposium on research in attacks, intrusions and defenses (RAID); 2012.

Mitsuaki Akiyama received the M.E. degree and the Ph.D. degree in Information Science from Nara Institute of Science and Technology, Japan, in 2007 and 2013, respectively. He has joined NTT R&D division in Tokyo from 2007, and he is now a member of NTT Secure Platform Laboratories. He has been engaged in research and development of network security, especially client honeypot and malware analysis.

Takeshi Yagi is a research engineer at NTT Corporation. He received his M.E. in Science and Technology from Chiba University. Since joining NTT in 2002, he has been engaged in research and design of network architecture, traffic engineering, honeypots, and security-data analysis based on machine learning.

Takeshi Yada is a Senior Research Engineer, Supervisor, Cyber Security Project, NTT Secure Platform Laboratories. He received the M.S. degree in engineering from Tokyo Institute of Technology, Tokyo, 1991. Since joining NTT in 1991, he has been engaged in research and development of network architecture, measurement and inference for network traffic, and network management.

Tatsuya Mori is currently an Associate Professor at Waseda University, Tokyo, Japan. He received B.E. and M.E. degrees in applied physics, and Ph.D. degree in information science from the Waseda University, in 1997, 1999 and 2005, respectively. He joined NTT lab in 1999. Since then, he has been engaged in the research of measurement and analysis of networks and cyber security.

Youki Kadobayashi received the Ph.D. degree in computer science from Osaka University, Osaka, Japan, in 1997. From 1997 to 2000, he was with the Computation Center of Osaka University as an Assistant Professor. Since 2000, he has been an Associate Professor with the Graduate School of Information Science, Nara Institute of Science and Technology, Nara, Japan.