Proceedings of the APAN – Network Research Workshop 2011

Analysis of Redirection Caused by Web-based Malware

Yuta Takata [1,*], Shigeki Goto [1,*] and Tatsuya Mori [2]

1 Waseda University / 3-4-1 Okubo, Shinjuku-ku, Tokyo, 169-8555 Japan
2 NTT Service Integration Laboratories / 3-9-11 Midori-cho, Musashino-shi, Tokyo, 180-8585 Japan

E-Mails: {y.takata, goto}@goto.info.waseda.ac.jp; mori.tatsuya@lab.ntt. co.jp

*Tel.: +81-3-5286-3182; Fax: +81-3-5286-3182

Abstract: Web-based malicious software (malware) has been increasing over the Internet. It poses threats to computer users through Web sites. Computers are infected with Web-based malware by *drive-by-download* attacks. Drive-by-download attacks force users to download and install the Web-based malware without being aware of it. These attacks evade detection by using automatic redirections to various Web sites. It is difficult to detect these attacks because each redirection uses the obfuscation technique. This paper analyzes the HTTP communication data of drive-by-download attacks. The results show significant features of the malicious redirections that are used effectively when we detect malware.

Keywords: Web-based malware; drive-by-download attacks; packet capturing.

---

1. Introduction

Damage resulting from Web-based malware has been increasing. Web-based malware uses a *drive-by-download* technique as its attack methods. Drive-by-download attacks force computer users to download and install malware without being aware of it by exploiting the vulnerabilities in a Web browser or some external components [1]. Figure 1 illustrates a typical drive-by-download attack. A user accessing an *entrance site* is redirected to malicious Web sites in sequence. These consist of three separate Web sites. A *zombie site* redirects the user to the next zombie site or an attack site. The zombie site is used as a stepping stone. An *attack site* exploits the vulnerabilities of the user's Web browser and forces the user to download malware from the *malware distribution site*, which contains malicious script codes or contents. These script codes

are difficult to analyze because they are often obfuscated. Therefore, it is not easy to detect zombie-site URLs, attack-site URLs, and malware-distribution-site URLs used in drive-by-download attacks.

There are two problems related to drive-by-download attacks. The first problem is that malicious Web sites attack users only when they access the malicious Web sites. This makes it difficult to detect the malicious Web sites because users only access them occasionally. The second problem is that a normal Web site may be compromised, causing it to play the role of an *entrance site* or a *zombie site* in drive-by-download attacks. An infected popular site like a social network service will impact a large number of users. The drive-by-download attack increases the risk to Internet users. Ensuring the security of the Internet poses a serious problem in daily life.

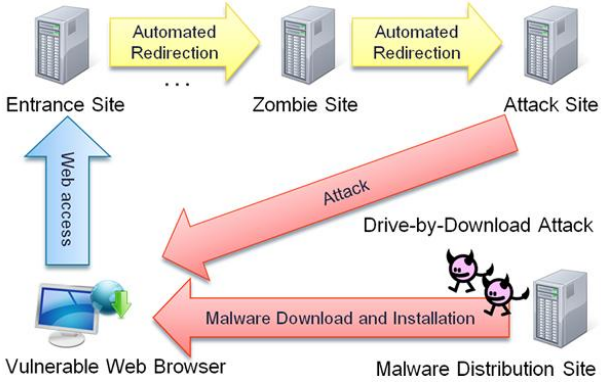

Figure 1. Drive-by-download attack.

There have been numerous research projects regarding drive-by-download attacks based on the measurement and analysis of malicious contents.

Egele et al. [2] illustrated and analyzed malicious JavaScript codes. They proposed building defensive mechanisms into a Web browser to mitigate the threats that arise from drive-by-download attacks. Their study showed a successful approach for mitigating drive-by-download attacks based on malicious script codes.

In this paper, we propose a new detection method for drive-by-download attacks using features of the malicious redirections. Our new method is not limited to JavaScript analysis because we observe a sequence of packets between a Web browser and servers.

Cova et al. [3] presented a method for the detection and analysis of malicious JavaScript codes. They developed a system that uses numerous features and applied machine-learning techniques to discriminate the characteristics of normal JavaScript code. Their system can identify anomalous JavaScript codes by emulating the behaviors and comparing them to the normal JavaScript profile. Their system specializes in JavaScript codes.

In addition to JavaScript codes, this paper covers other features such as HTTP methods and URL information. It should be noted here that the evaluation of JavaScript codes is very expensive. Therefore, although we consider only the existence of JavaScript codes, our method does not analyze the code in detail.

2. Newly Proposed Method

This paper proposes a new method for finding the hidden malicious URLs in drive-by-download attacks by analyzing redirections from captured HTTP communication data packets. It is relatively easy to trace redirections in HTTP communication by looking for the *referrer (or referer)* fields in GET requests and HTTP responses. However, JavaScript codes can hide a referrer field in the malicious redirections of drive-by-download attacks. This is why a simple approach, like a blacklist of malicious URLs, does not work effectively. In fact, we find that the occurrence of a referrer field in malicious redirection accounts for only 10% of malicious attacks. In this paper, we analyze the HTTP communication data captured in controlled environment where only the drive-by-download attacks exist. We describe this data-capturing environment later. We try to reveal the features of the redirection to make it possible to detect unknown malicious attacks effectively.

2.1. Taxonomy: Session and Server

We define a *session* as a chain of packet flows. It starts from a DNS name resolution, continues with an initial three-way handshake in TCP followed by HTTP communications, and finally finishes with FIN or RST packets in TCP. We can trace a session by sorting with SEQ/ACK numbers in TCP using the same MAC address, IP address, and port number.

We next define a *server*. A server is identified by an IP address and domain name[1]. When we access a Web site, we connect to a Web *server* or *servers*, and the Web browser establishes sessions with the Web *server* or *servers*.

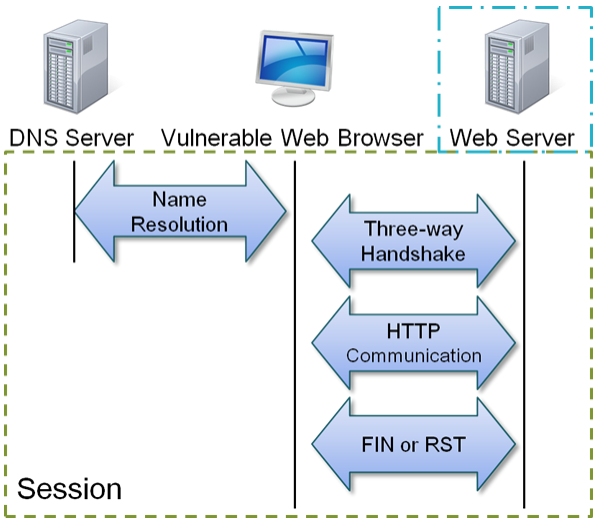

Figure 2. Session and server

[1] We need to use an IP address and domain name pair because we can operate other domain names with the same IP address by using the DNS *round robin* and the *virtual domain* mechanism.

2.2. Test Range

Web browsers have a *progressive rendering* function that accelerates the rendering of a Web page. This function evaluates data such as HTML files and JavaScript files immediately after the download. For this reason, we start by analyzing the data from GET requests and define the test range. The three above mentioned test methods are applied to GET requests and the HTTP responses generated from the GET requests. HTTP responses are analyzed for 120 seconds time intervals. We defined this time interval empirically using a preliminary experiment that measured the time interval from the generation of redirection to the download of the first malware. Therefore, this range depends on the network environment, including the bandwidth and number of connected users.

2.3. New Method 1: URL Test

Our new method consists of three tests for detecting redirections. The first test investigates URLs. The URL *test* detects redirections by picking up URLs from an HTML content file and a location field from an HTTP response header. Thus, the URL test covers two methods.

The first method analyzes all of the characters in an HTML content file and extracts URLs. We assume that all of the HTML files targeted for analysis are from an *entrance* site because we are only analyzing the communication of drive-by-download attacks. In addition, we assume that all of the following HTTP communications will be made automatically by the attack. We look for GET requests to the above mentioned URL list. When the URL list has GET requests, we consider that these requests were generated by the redirection.

The second method gets a URL from the *"location"* field in an HTTP response header. At this point, the HTTP response header includes the HTTP status code "3xx," which indicates a redirection to the URL set in the location field.

2.4. New Method 2: Referrer Test

The referrer *test* detects redirections based on the referrer field of the GET request. If the previous URL is set in the referrer field of the GET request, we can trace the redirection. When we move to a newly clicked URL, the origin site URL is set as the referrer field.

2.5. New Method 3: Host Test

The above mentioned URL test and referrer test are straightforward methods for detecting redirections. However, these methods cannot detect redirections when URLs are obfuscated or no referrer redirection is generated by JavaScript. In this paper, we solve these problems by using the host *test*, which analyzes the servers and sessions.

First, we collect the list of all of the servers and sessions in the communication data. We call this list the *first* list. Next, we analyze the communication data again and perform the URL test (2.3) and referrer test (2.4) for each entrance Web site. Some servers and sessions are found to be redirected. We put the redirected servers and sessions into the other list. We call the list the *second* list. Finally, if any sessions in the first list are not covered by either the URL test or referrer test, we investigate the remaining HTTP communications in detail. If there are any sessions in the first list with the known servers that appear in the second list, these sessions are classified as redirected sessions and belong to the same redirection group as the URL test and referrer test.

The host test is based on the fact that many Web pages are managed by a small number of servers called *hosts*. Figure 3 shows an example of a *host* test.

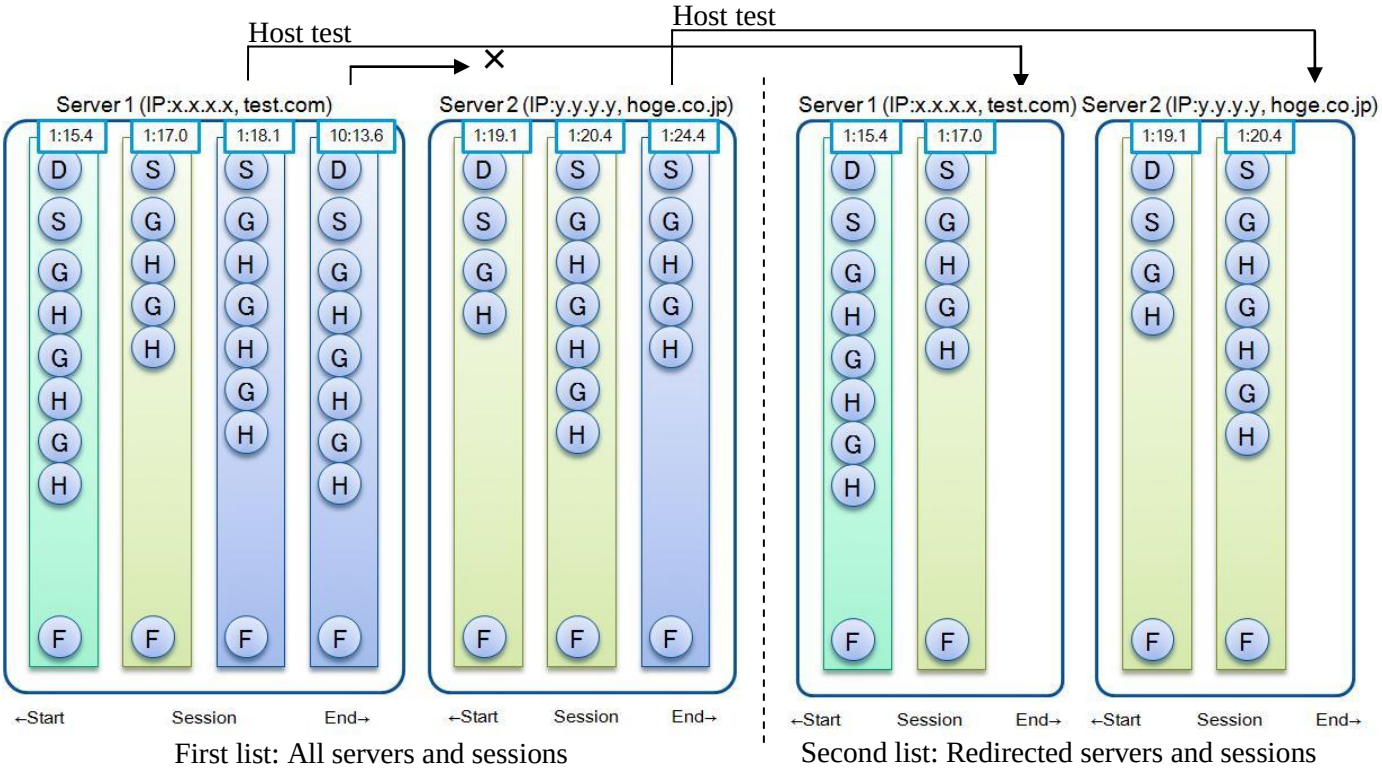

Figure 3. First list and Second list

Each vertical rectangle shows a session. A session contains a DNS name resolution (D), 3-way handshake (S), HTTP communication (G, H), and FIN or RST packets (F). Each session indicates the first packet arrival time of the session at the top of the rectangle. 1:15.4 means 1 minute and 15.4 seconds past from the origin of the timer.

The leftmost session at 1:15.4 is generated by a user who click an entrance Web site URL. If URL test or referrer test identifies sessions at 1:17.0, 1:19.1, and 1:20.4 as sessions generated from the entrance Web site, then the sessions identified in the *first* list are put into the *second* list in Figure 3.

In Figure 3, sessions at 1:18.1 and 1:24.4 are not identified by URL test or referrer test because of obfuscating. These sessions started within 120 seconds from the timestamp of the entrance session and they communicated with the same server. Therefore, the *host* test classifies these sessions into the same redirection group. However, the session at 10:13:6 is not classified into the same redirection group because the session started about 10 minutes later from the redirection and out of the test range of 120 seconds (2.2).

We found redirections by three tests from the *first* list, and put the result in the *second* list.

2.6. Malware Detection

The purpose of a drive-by-download attack is to force a victim to download the malware. Therefore, when a Web browser downloads executable files, we can determine that these redirections are malicious. We used the data captured under a controlled environment containing a collection of drive-by-download attacks. The data was provided by the Malware Workshop 2010 Datasets project in Japan to facilitate data analysis in the security research area [4]. In this study, we used the D3M 2010 datasets in the MWS 2010 datasets. The D3M 2010 datasets contain accessed URL lists, which are found on a public URL black list [6], along with previously detected drive-by-download attacks. They collected malicious communication data using a Web client honeypots called Marionette [5]. Marionette does not execute the downloaded malware, even though it comes under attack because of its vulnerabilities. The D3M 2010 datasets were captured on March 8th, 9th, and 11th of 2010. We can observe the detection performance for the malicious redirection by successfully detect the download of executable files using the three proposed test methods.

3. Results of Experiments

We implemented the program to evaluate the performance of our newly proposed methods.

3.1. Program Execution Example

Figure 4 shows a portion of the results from the program.

```
54:http://****ting.com/webalizer/050709wareza/crack=17=keygen=serial.html
   79:http://****ting.com/webalizer/050709wareza/images/5.jpg
   83:http://****ting.com/webalizer/050709wareza/images/6.jpg
   91:http://****ting.com/webalizer/050709wareza/images/8.jpg
   122:http://****ting.com/webalizer/050709wareza/images/1.jpg
   132:http://****ting.com/webalizer/050709wareza/images/2.jpg
   160:http://****ting.com/webalizer/050709wareza/images/7.jpg
   165:http://****ting.com/webalizer/050709wareza/images/3.jpg
   176:http://****abie.in:3129/js
   210:http://****eegh.in:3126/download/index.php
      248:http://****eegh.in:3126/download/jabber.php
759:('exe(Inline)', 'octet-stream') http://****eegh.in:3126/download/banner.php?spl=mdac
```

Figure 4. Program execution.[2]

In Figure 4, the first line indicates the 54th packet to obtain the HTML file. Subsequently, the Web browser obtains seven JPEG files from the same domain, and the next part refers to other domain files. We can see that the 176th, 210th, and 248th Web pages are zombie sites because these URLs contain suspicious characters such as "download" and use ephemeral port numbers rather than the port number (80). The indent in Figure 4 represents a reference or a redirection identified by using the URL test or the referrer test. The contents from 79 to 210 are generated from the 54th HTML file. Similarly, the 248th PHP file is generated from the 210th PHP file. Finally, the Web browser accesses the file called "banner.php", and we confirm the information from the HTTP response header fields such as "exe (Inline)" and "octet-stream". We can thus identify it as an executable file. This means the 759th line indicates a redirection, and it is classified into the same redirection group as the host test. We cannot identify the origin site of this redirection.

Figure 4 also shows that the Web browser accesses various Web sites (domains) and downloads an executable file (malware) by merely accessing an HTML file.

3.2. Test Performance

We counted the success rate for each test to acquire the malware distribution URLs. The results are listed in the following table.

Table 1. Tests to acquire malware distribution URLs.

Date	Marth 8th	March 9th	March 11th
Total of malicious URLs	202	205	158
URL test	12 (5.9%)	10 (4.9%)	10 (6.3%)
Referrer test	13 (6.4%)	13 (6.3%)	6 (3.8%)
Host test	177 (87.6%)	182 (88.8%)	142 (89.9%)

[2] We masked these URLs with "*" for security.

3.3. Using Obfuscated JavaScript

We discovered that obfuscated JavaScript codes are used as entrance sites and zombie sites with a probability of 100% in the D3M 2010 datasets. The obfuscated JavaScript codes are used not only for HTML files but also for PDF files.

We discovered objects that are used frequently in obfuscated JavaScript codes for example, the `eval` function, `String` object's functions (`fromCharCode`, `replace`, `split`, and so on), and `location` objects.

3.4. HTTP Header Parameters: Referrer Field, Server Field, and Vary Field

By analyzing the HTTP response header parameters, we discovered three significant features of malicious redirections. First, most redirections to malware distribution sites are not simple referrer redirections to hide the attack origin site. The attackers use JavaScript codes and force Web browsers to download malware with no explicit referrer. Second, some malware distribution servers use Nginx rather than Apache for the Web server. Third, a number of HTTP response headers set "User-Agent" in the Vary field[3].

3.5. URL Features

Attackers use specific parameters for GET variables. They use GET variables such as "sql" and "mdac" and set the user's environment variables. One example is the URL "http://hoge.com?sql=2&br=MSIE&vers=6.0". Additionally, sometimes an IP address and ephemeral port are also used in a variable, like the 759th packet in Figure 4.

3.6. Using Fake Files

We investigated "Content-Type" fields and counted these fields at the HTTP responses of the downloaded malware. We discovered malware and attack scripts faked to image files.

3.7. Features Utilization Ratio

We obtained the data listed in the following table after analyzing the usage rates for the above mentioned features in D3M 2010.

[3] When the contents are changed for the User-Agent, the Vary field is set to "User-Agent"; for example, "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)" in the D3M 2010. User-Agent is a field that is used to distinguish the model of the browser or mobile.

Table 2. Counts of redirections using features in D3M 2010.

Date	March 8th	March 9th	March 11th
Total of malicious redirections	98	89	77
Obfuscated JavaScript	98 (100%)	89 (100%)	77 (100%)
No referrer redirection	89 (90.8%)	79 (88.8%)	75 (97.4%)
PDF + JavaScript	35 (35.7%)	33 (37.1%)	25 (32.5%)
URL Feature	38 (38.8%)	30 (33.7%)	24 (31.2%)
Vary: User-Agent	29 (29.6%)	27 (30.3%)	25 (32.5%)
Server: Nginx	29 (29.6%)	24 (27.0%)	21 (27.3%)
Ephemeral Port	8 (8.2%)	5 (5.6%)	10 (13.0%)
Fake files	7 (7.1%)	6 (6.7%)	4 (5.2%)

Table 2 indicates that the use of obfuscated JavaScript codes with no referrer redirection can be used as the *fingerprints* of malicious redirections because these features are used 100% of the time. Additionally, Table 2 implies that the use of an ephemeral port and fake files are significant features because normal Web sites do not use them.

3.8. Unique IP Address in Redirection

Figure 5 shows the number of unique IP addresses in each redirection. It also shows that a single IP address has been used heavily. This means attackers often operate a number of domains with a small number of IP addresses using a virtual domain. Such a feature is also an effective criterion to detect malicious redirection.

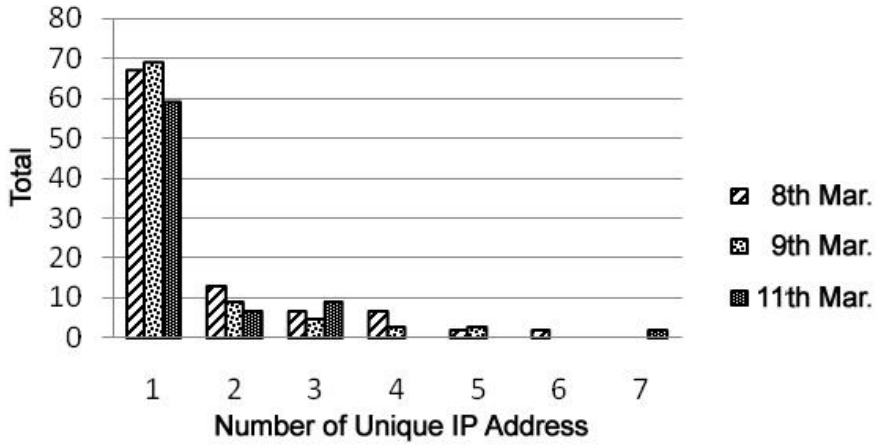

Figure 5. Number of unique IP address in redirections.

4. Conclusion

We analyzed communication data captured in an environment where only the communication data of drive-by-download attacks existed. We found the significant features of malicious redirection. The new methods successfully detected the redirections by using these features. Our future plans are to evaluate the extraction method of malicious redirections by using the acquired features from normal and malicious communication data and to apply the proposed methods to communication data captured in various networks.

Acknowledgements

We are thankful for the MWS 2010 datasets provided by the NTT Communications Corporation, as well as for our discussions with NTT Information Sharing Platform Laboratories.

References

1. Moshchuk, A.; Bragin, T.; Gribble, S.D.; Levy, H.M. A crawler-based study of spyware on the Web. *University of Washington*, 2006; Vol. *2*.
2. Egele, M.; Kirda, E.; Kruegel, C. Mitigating drive-by download attacks: challenges and Open Problems. *INETSEC 2009 IFIP AICT 309***;** Vol. *4*, pp. 52-62.
3. Cova, M.; Kruegel, C.; Vigna, G. Detection and analysis of drive-by-download attacks and malicious JavaScript code. *WWW 2010*; Vol. *4*, pp. 281-290.
4. Hatada, M.; Nakatsuru, Y.; Akiyama, M.; Miwa, S. Datasets for anti-malware research ~MWS 2010 datasets~. *Computer Security Symposium 2010***;** Vol. *10*, pp. 19-21.
5. Akiyama, M.; Iwamura, M.; Kawakoya, Y.; Aoki, K.; Itoh, M. Design and implementation of high interaction client honeypot for drive-by-download attacks. *IEICE Transactions on Communications*, 2010; vol. *5*, pp. 1131-1139.
6. Malware Domain List. *http://malwaredomainlist.com*
7. Takata, Y.; Mori, T.; Goto, S. Redirect analysis of web-based malware. *The 73rd National Convention of IPSJ*, 2011; Vol. 3.