Malicious Website Detection Based on Honeypot Systems

Tung-Ming Koo, Hung-Chang Chang, Ya-Ting Hsu
Department of Information Management
National Yunlin University of Science & Technology
Douliou, Yunlin, Taiwan, R.O.C.
e-mail: {koo, g9823811, g9723742}@yuntech.edu.tw

Huey-Yeh Lin
Department of Finance
National Formosa University
Huwei, Yunlin, Taiwan, R.O.C.
e-mail: linhytw@nfu.edu.tw

***Abstract*—In the Internet age, every computer user is likely to inadvertently encounter highly contagious viruses. Over the past several years, a new type of web attack has spread across the web, that is, when a client connects to a malicious remote server, the server responds to the request while simultaneously transporting malicious programs to the client's computer, thereby launching a drive-by download attack. If the attack is successful, malicious servers can control and execute any program from the client's computer. Malicious websites frequently harbor obfuscation mechanisms to evade signature-based detection systems. These obfuscators have become increasingly sophisticated that they have begun to invade multimedia files (JPG, Flash, and PDF). Under such circumstances, unless specific behaviors are triggered by malicious webpages, identifying programs with malicious intent by merely analyzing web content is extremely difficult, not to mention the formidable quantity of webpages and the ever changing attack techniques. Based on a client-side honeypot system, this study proposes a model for determining whether a webpage is malicious. We present a technique to improve the accuracy of malicious web detection. First, static content analysis is performed to accelerate the detection, followed by actual browsing on webpages for in-depth probing using the client-side honeypot system. Using this method, user's security is protected when surfing the Internet.**

Keywords- Honeypot; malicious website; drive-by download

I. INTRODUCTION

A web browser is indispensable tool for browsing online content. However, a security report by IBM published in the first half of 2009 [10] indicated that web browser vulnerabilities have been the most exploited over the past few years. Regarding the range of operating systems, the Microsoft operating systems family was ranked first place for the severity and incidence of system breaches. Furthermore, Microsoft has the largest user base. Therefore, this study uses Microsoft operating systems to build the experimental environment.

When a client connects to a malicious remote server, the server transfers malicious programs to the client's computer. Then, when the user sends a request to the malicious server, the server responds to the request while simultaneously launching a drive-by download attack[17]. If the download is completed successfully, malicious servers can execute any programs on the client's computer. According to statistics by the Symantec Corporation [5][17], more than 18 million drive-by downloads were enforced in 2008.

Several reasons can be attributed to why malicious attacks use the web as the medium for attacking. First, the HTTP protocol is very easy to configure and use. Second, using a generic communication protocol is inconspicuous. Because most firewalls permit port 80 of the HTTP protocol to pass, they are often useless to withstand such attacks, allowing attacking packets to invade the system effortlessly. Because interconnectivity has become an indispensable part of everyday life, terrorists believe that attacking through the web is an ideal option because of its maximum benefits and minimum risks.

Malicious websites frequently use obfuscation techniques to evade detection; these obfuscations have become increasingly sophisticated, even extending to multimedia documents (JPG, Flash, and PDF). Under these circumstances, unless specific behaviors are triggered by browsing malicious webpages, identifying programs with malicious intentions by analyzing only the content is extremely difficult, not including the daunting quantity of webpages and constantly changing attacking techniques. Thus, this study first analyzes the static content of webpages to accelerate the analyzing process; then, we use a honeypot system to browse webpages for in-depth probing. The analysis outcome is used to determine whether webpages are benign or malicious automatically.

The followings are the main objectives of this technique:

- Active detection of malicious websites
- Automated malicious website analysis
- Establish a protective mechanism

II. LITERATURE REVIEW

Traditional attacks are launched from the server-side, that is, the attack and penetration are aimed at the server service system. Nowadays, attacks have been shifting to the client-side, that is, when a client-side application (browser) interacts with malicious servers, the vulnerabilities of the application are exploited for malicious purposes, or the client-side is induced to execute malicious programs. Any client/server architecture can lead to this type of attack (Web, FTP, and Mail), which is difficult to detect with the current firewall, invasion detection, and proxy-based defense systems.

Drive-by downloads[1][17], also called "forced downloads" or "pass-by downloads," are a new type of client-side attack. When a user visits a webpage containing malicious scripts, the malicious programs are downloaded to the user's computer without the consent of the user. Even

legitimate websites can carry drive-by downloads simply because they were compromised and injected with malicious scripts. If the web browser or other software is not patched or updated regularly, the user may become infected.

The process of drive-by downloads are as follows:

1. Attackers seize control of legitimate web servers
2. Web browsing by users
3. Drive-by downloads
4. Redirecting to malicious websites
5. Malicious website attacking the user

Numerous studies have investigated malicious webpage detection and protection in recent years. Static analysis examines whether the program code or original documents carry the defined signatures or patterns without executing these programs. Dynamic analysis refers to the detection technique that uses behavioral patterns to simulate the browsing environment of users by loading an actual webpage into the browser and determining whether the browser downloads or executes unwanted scripts.

A honeypot is a type of information system resource that can be used without authorization or illegally. Thus, honeypots are primarily deployed as targets to be detected, attacked, or harmed by exploitive code. The basic assumption is that because honeypots simulate non-existent systems or services, a person attempting to reach the non-existent systems or services must have bad intentions.

Most honeypots are server-side systems. They are configured as vulnerable hosts providing specific services to ensure they are detected and assaulted by hackers and damaged by malware. Honeypots are attacked passively and do not act as a trapping system without first being attacked.

Over the past several years, with the attack targets shifting to the client side, Spizner developed a new client honeypot type [4]. This honeypot differs from the traditional server honeypot in that its active sniffing creates interactions with the target hosts, and it is designed to actively detect offensive behaviors from malicious hosts. By contrast, server honeypots cannot sniff actively. Therefore, a client honeypot is used to actively detect whether a web service provided on the Internet contains malicious behaviors.

Microsoft previously conducted HoneyMonkey, a project designed to analyze the existence of malware in a particular website [9]. Run on a Windows XP operating system, HoneyMonkey sends queries to webpages and waits to determine whether they launch attacks against the Windows system. Additionally, HoneyMonkey records invasions by hackers or attacking behaviors by worms in an attempt to protect the Windows system from attack. First, HoneyMonkey runs a Strider Flight Data Recorder (FDR) program to monitor every file in the directory and the read-write behaviors of the system registry. Then, it launches a web browser to view a webpage on a certain website and waits several minutes on every page it visits. Meanwhile, HoneyMonkey does not accept any requests from the dialog windows requesting installation. When browsing the webpages, all executable files not established in the temporary directory are recorded by the FDR. Using this approach, HoneyMonkey determines whether a website contains malicious code. However, this approach only assesses through manual work and only analyzes specific websites.

Honeyclient is a highly interactive web-based honeypot developed by Wang in 2004 [12]; it was the first open source client honeypot written using Perl. As an event-based application, Honeyclient can detect attacks at the client side by monitoring specific directories and the system registry. Using the password hash MD5, Honeyclient compares the directories and registry after interacting with the server. When the checksum differs from the original, it is considered malicious. However, this approach is time-consuming and has a high false positive rate.

In 2008, Seifert proposed Capture [3][13], a highly interactive client honeypot with full functionalities. This honeypot uses a virtual machine to simulate the system environment at user's end. By controlling the browser, it detects the remote target hosts and observes changes in the system status. Modifications to the system status may suggest an abnormality. Although this approach is precise, detection is time-consuming.

III. Research Methods

Thousands of drive-by downloads occur on mainstream websites daily; the users of these websites are infected or attacked at an astonishing rate. Unfortunately, numerous users remain unaware of these attacks. Thus, this study investigates using static content analysis and dynamic behavior analysis to determine whether a webpage has malicious or abnormal intentions. We employ semantic-based static content analysis and the behavior capturing technique proposed by Seifert et al. [2][14][15], and attempt to improve these methods considering the network characteristics.

Currently, most malicious webpages are assessed using static analysis or signature-based value. Despite the provision of rapid detection and judgment, these methods cannot resist unknown attack variations. Although heuristics and behavior-based detection techniques can prevent unknown attacks, they are disadvantaged by being time-consuming. Conventional antivirus software, firewalls, and invasion detection programs are totally useless when confronted with the flexibly adapting and swiftly concealed attacking tactics. In response, this study proposes a detection and defense model that combines static content analysis and dynamic behavior analysis to develop an actively detecting, dynamically analyzing, and rapidly defending mechanism.

A. System Architecture

As shown in Fig. 1, the system architecture of this study comprises four modules, that is, a proxy module, source code analysis module, behavior recording module, and behavior analysis module, as Fig. 2. The webpage analysis portion adopts a hybrid system, where static content analysis is used to improve detection efficiency, and dynamic behavior analysis is used to ensure the extensiveness of the detection. The source code analysis module is part of the static content analysis stage; whereas the record behavior module and the behavior analysis module are part of the dynamic behavior analysis stage.

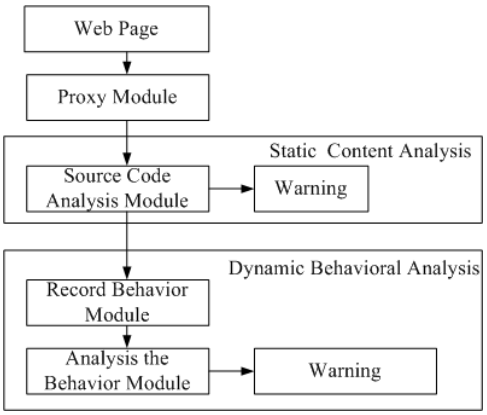

Figure 1. System architecture

B. System Modules

(1) Proxy module： The proxy module is primarily responsible for protecting the clients of the internal network. This module filters the webpages accessed by the clients, records the URLs, saves the webpages, and then sends a copy of this page to the static analysis module. This module waits for the analysis results before deciding whether to transmit the webpage to the client side. The results of this analysis are recorded to enable the server to respond immediately if the same webpage is requested again.

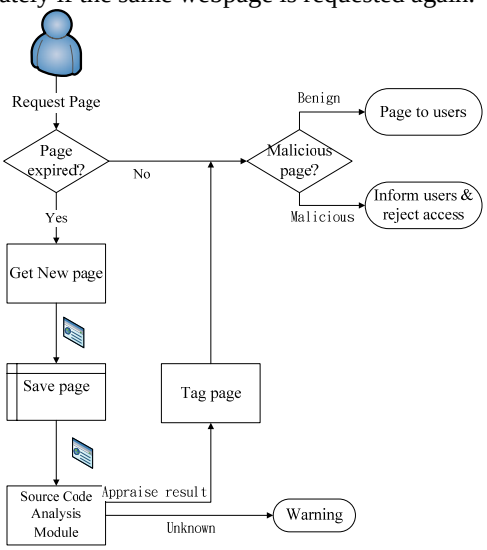

Figure 2. Flowchart of the proxy server module

The webpage detection cycle adopts a mechanism similar to the transparent proxy (TP). The difference between a TP and other proxy servers is that, when using a TP, no browser settings are required. That is, the clients can redirect any HTTP connections to the proxy server without changing any settings. The rules used in this study to determine the validity of the detection results are as follows:

1. If the detection time on a webpage exceeds the life cycle of that page, the detection is expired; otherwise, it is a valid detection. The life cycle of the webpage refers to the parameter set by the web server.
2. If the detection time on a webpage exceeds the time set for the maximum parameter, the detection is expired.
3. If the detection time on a webpage exceeds the last modified time factor of the document, the detection is expired; otherwise, it is a valid detection. The last modified time factor of the document refers to the time the page is cached minus the last modified time the webpage was cached on the web server * the percentage set.
4. If the detection time on a webpage is shorter than the time specified by the minimum parameter, the detection is valid.
5. If none of these conditions are met, the detection is expired.

(2) Source Code Analysis Module： This module uses static content analysis to test the original HTML syntax according to identified pattern values. This module is based on abnormal semantics analysis. First, regarding the obfuscation features, the obfuscation techniques in the webpage script are recorded for further identification and pattern matching of the URL. Generally, the URL does not contain numerous bytes; thus, the speed of identification can be improved. Next, the static pages are filtered. The purely static webpages free of scripts and automatic link tags are filtered first to accelerate the detection rate and because they are considered harmless webpages. Many web application attacks or leakage attacks are conducted by exploiting the script syntax, or through automatic links that connect the user to malicious image files, reducing the probability of detection. Finally, the source code of the webpage is examined to determine whether abnormal semantics are present. This determination requires the source code of the page. After the three major assessments, if the page cannot be identified as either good or malicious, the URL of the webpage is sent to the behavior recording module for dynamic behavior analysis.

This module comprises four submodules, which are described as follows:

1. Obfuscation signatures: This submodule primarily records the webpage scripts that contain obfuscating features for later analysis.

2. URL Signature: Filters the malicious semantics using the suspicious URL signatures.

3. Static Webpages Assesses: Whether the webpages contain syntax that performs syntax calls (including JavaScript, VB Script, etc). Numerous web application attacks and system breaches are accomplished by exploiting the syntactic scripts. The most common method to determine whether a webpage contains provocative scripts is to check whether any <script> tag is in use. Another screening criterion is whether it contains automated links. Automated links refer to when the webpage links to other webpages or accesses information from other webpages without the user's consent, or the user is unaware of the linkage. Hackers may embed malicious URLs as automatic linking tags in the webpages, and the tags do not necessarily comply with the

syntax. A number of HTML tags are exploited by hackers to create automatic linking tags, such as Iframe, Meta, Img, Embed, Xml, Style, Object, and Applet. The criterion for the final judgment is whether it can call other scripts or contains tags with automated links. If neither is true, then it can be judged a normal static webpage. This step identifies portions of the normal webpages first to accelerate detection. However, if either criterion is true, or both are true, then the webpage can be judged suspicious and should undergo further examination in the following phases.

4. Signatures in the source code. Following static content analysis, if a webpage is identified to have any of the malicious signatures, it can be regarded as a malicious webpage immediately. If the signatures are not sufficiently decisive enough for judgment, the URL can be sent to the behavior recording module for behavioral analysis. Because behavioral analysis is relatively time-consuming, the system responds to the user first, notifying them that the webpage is suspicious; however, it is still accessible if the user insists.

(3) Behavior Recording Module： This module primarily operates on Capture-BAT, a client honeypot [4] installed on a virtual machine. This module browses webpages in the simulated Windows client-side environment and records all events triggered when viewing the webpages, including monitoring the I/O operations of the file system, changes in the registry, and creation and destruction of processes. Then, the regular events that are triggered by the operating system or the browser itself are excluded and the recorded document is sent to the behavior analysis module for assessment.

This module comprises three submodules, which are described as follows:

1. Simulated browsing

Each detection starts by visiting a website in a clean environment to ensure the changes in the system are caused by the events that occur when browsing the webpages. Before proceeding to visit the next website, the simulated environment is restored to its initial clean status.

2. Recording the triggered events

API hooking is used to monitor changes of system status, including the following:

- The I/O operations of the file system. Monitor the reading and writing of all the documents and record the time, type of action (Read, Write), triggering program, and the full path of the documents.
- Changes in the registry. Monitor modifications to the Windows system configuration and record the time, type of action (setValKey, DeleteValKey…, etc.), triggering programs, and full path.
- Creation and closing of processes. Monitor the creation and destruction of processes, excluding the programs already running before browsing was initiated. Record the time, type (Created, Destruction), triggering programs, and the full file path.

3. Excluding normal events

Under normal conditions, events occur constantly in the system; therefore, the normal events must be excluded to prevent false positives. Exclusion lists are set for the file system, registry, and processes. The plus sign denotes normal events that should be excluded, and the minus sign denotes abnormal events that should not be excluded. In Table 1, the data in the second and third row indicate that the event writing C:\WIN\ can be excluded, whereas the event writing C:\WIN\Sys32 should not be excluded.

Table 1. Example of the excluding list for the file system

Excluding	Type	Program Name	Category of Document
+	Read	.*	.*
+	Write	.*	C:\WIN\
-	Write	.*	C:\WIN\Sys32\
+	Write	C:\\Browser.exe	C:\Cache\

After this module, the document recording the events triggered when viewing the webpage is transmitted to the behavior analysis module for analysis.

(4) Behavior Analysis Module： Analyzes various types of events according to the records of webpage browsing using the client honeypot. In this module, the criteria for judgment are based on the impact or effects each event has on the system.

In this module, various system changes resulting from the events triggered when browsing webpages are analyzed and scored according to the severity of damage. This module analyzes three major events.

1. Analysis of file I/O events. Discriminating if there are file I/O operations according to the recorded document. If a document is accessed without authorization, it is considered malicious behavior. Malicious signatures may include modifying documents under C:/windows/sys32/ and writing to documents that contain macros for automatic execution.
2. Analysis of registry events. After installation in the system, all applications are registered to the registry to ensure the usability of the application. If a webpage modifies the registry, it is highly likely it contains a hazardous link. The link is then analyzed by the system according to the modifications it made to determine whether it is a malicious link.
3. Analysis of process events. When executing a normal application, the processes of the application can be observed clearly. Information such as the CPU status and memory usage can be observed from the system process, and which application initialized a particular process is also clearly displayed. When a connection to the webpage is established, a review of the process record can reveal the occurrence of abnormal disconnection in the programs, shifts from a regular process to a resident process, or whether processes are concealed using Rootkit, thereby determining whether the connection is a malicious one.

Finally, if any malicious signatures are identified through analyses of the three types of events, the webpage can be considered a malicious one.

IV. SYSTEM IMPLEMENTATION AND EXPERIMENTATION

For the static analysis process, we used a self-developed JAVA program to process the signatures with regular expressions to accelerate the matching speed. For the

dynamic analysis process, we used a honeypot system named Capture-HPC [5], where the excluding list and conditional constraints are incorporated to achieve optimal performance. The behavior recording module was implemented under a VM environment, which sequentially simulates website browsing according to the order of the URLs requiring detection. The browsing history was recorded fully, and the malicious packets in the network traffic were collected using Wireshark.

A. Sample of the Experiment

During the four-month period, approximately 1,000 URLs were collected, of which, 477 were malicious and 598 were good. The majority of the benign URLs were obtained from the top 500 global websites [8]; a number of them were sourced from a list of secure websites known to web crawlers by searching using Google [11]. To ensure consistency of the data, all the sample webpages were obtained using Flashget and archived after the validity of the URLs was confirmed. Subsequent experiments were conducted based on these sample webpages. All samples were scanned using Avira AntiVir personal edition. No viruses were found in the benign sample, whereas in the malicious sample, 213 virus warnings were provided and 52 varieties of virus were found, as shown in Table 2.

Table 2. Virus codes contained in the sample

BDS/Backdoor.Gen back-door program	TR/Agent.GY.965 Trojan
BDS/IRCNite.HF back-door program	TR/Click.Agent.ldb Trojan
EXP/Pdfka.jgw exploit	TR/Clicker.nhi Trojan
EXP/Pidief.bwf exploit	TR/Crypt.CFI.Gen Trojan
EXP/Pidief.dbw.7 exploit	TR/Crypt.FKM.Gen Trojan
HTML/Dldr.A.aqq.9 HTML script virus	TR/Crypt.XPACK.Gen Trojan
HTML/ExpKit.Gen HTML script virus	TR/Dldr.Agent.diau.6 Trojan
HTML/FakeAV.down HTML script virus	TR/Dldr.Agent.fig.3 Trojan
HTML/IFrame.aaa HTML script virus	TR/Drop.Mul.1 Trojan
HTML/IFrame.akb.1 HTML script virus	TR/Drop.Raysun.A Trojan
HTML/Infected.WebPage.Gen HTML script virus	TR/Dropper.Gen Trojan
HTML/Script.INM HTML script virus	TR/FakeRean.A.580 Trojan
JAVA/ClassLoade.I.1 Java virus	TR/FraudPack.awch Trojan
JS/Agent.6795 JavaScript virus	TR/Gendal.122880.H Trojan
JS/Cosmu.A JavaScript virus	TR/Hijacker.Gen Trojan
JS/Dldr.Agent.15067 JavaScript virus	TR/Pakes.BR.1 Trojan
JS/Dldr.Agent.fig.2 JavaScript virus	TR/PCK.Tdss.Z.4764 Trojan
JS/Pegel.45700 JavaScript virus	TR/PCK.Tdss.Z.4781 Trojan
JS/Redirect.9200 JavaScript virus	TR/Ransom.XBlocker.abh Trojan
JS/Redirector.k.795 JavaScript virus	TR/Scar.cdxw Trojan
WORM/IrcBot.96396 worm	TR/Siscos.MZ.3 Trojan
WORM/IrcBot.96401.1 worm	TR/Spy.Agent.bfnn Trojan
TR/Agent.11776.U Trojan	TR/Spy.SpyEyes.GA.2 Trojan
TR/Agent.AO.1233 Trojan	TR/Spy.ZBot.afng.3 Trojan
TR/Agent.AO.1314 Trojan	TR/Swisyn.aedm Trojan
TR/Agent.AO.1315 Trojan	TR/TDss.bdfm Trojan

B. System Experiments

The system experiments were conducted in two forms, and the performance and accuracy of each experiment was compared.

a. Static analysis alone: All samples were analyzed using only static analysis, and the samples without malicious signatures were considered harmless.

b. Dynamic analysis alone: All samples underwent dynamic analysis, and the outcomes of the interaction were directly analyzed.

(1) Static analysis： All samples without malicious signatures were considered harmless. The URL signatures are then compared because URL addresses do not contain numerous bytes, therefore, the assessment can be accelerated. Subsequently, during the static webpage filtering process, purely static webpages with no scripts or tags of automated links are filtered first; most are immediately regarded as harmless webpages, which accelerates the detection process. The reason numerous web application attacks or leakage attacks are conducted by exploiting the script syntax, or through automatic links that connect to malicious image files, is to reduce the probability of detection.

Finally, the source code of the webpage is examined to determine whether abnormal semantics are present; this determination requires the source code of the page. The number of signatures in the samples is shown in Table 3. Among these signatures, the most prominent are the URL signatures. All items in the field “mismatch of URL document type and content” are clearly of a deceitful nature. Altogether, 147 instances were found in the malicious sample, whereas no instances were found in the benign sample. Similarly, instances of “syntax instructions following the URL” were only found in the malicious sample. These statistics enable the discrimination of malicious sample simply using static analysis without detecting the source code, saving a significant amount of time. Regarding the source code signature, obvious outcomes were observed. Signatures under the three headings “using base 64 decryption,” “keyword transcoding or splitting,” and “system variable alteration” only appeared in the malicious sample; none appeared in the benign sample.

Table 3. Instances of signatures from static analysis

	Static signatures	Malicious 477 instances	Benign 598 instances
Obfuscation signatures	Encrypted	207	160
URL signatures	Syntax instructions following URL	21	0
	Mismatch of URL document type and content	147	0
Static webpage	Containing syntax that can call scripts	267	544
	Containing tags of automatic linking	273	548
Source Code Signature	Redirection to different domains without warning	78	26
	Hidden page components	114	96

	Static signatures	Malicious 477 instances	Benign 598 instances
	Only one line in the js document	129	412
	Using base 64 decryption	9	0
	Keyword transcoding or splitting	198	0
	System variable alteration	6	0
	Containing advertising signatures	45	264

This study sets the four signatures "mismatch of URL document type and content," "redirection to different domains without warning," "keyword transcoding or splitting," and "system variable alteration" as definite malicious signatures because all four signatures feature malicious deception and three only appeared in the malicious sample. "Redirection to different domains without warning" also appeared 26 times in the benign sample, mainly because of the mutual URL forwarding between the enterprise website (such as Microsoft) and its affiliated services (such as Hotmail). Because involuntary user behavior or behaviors occurring without user awareness was defined as malicious behavior in this study, presenting a few misjudgments during detection is acceptable. The analytic result of static analysis alone is shown in Table 4. The results shows that 122 (96 + 26) false positives occurred, with an accuracy rate of 88.65%. The total time consumed was approximately 27 min.

Table 4. Number of judgments for static analysis

Result of the judgment	Malicious sample 477 entries	Benign sample 598 entries
Malicious	381	26
Benign	96	572

(2) Dynamic analysis alone： The outcome after interaction is analyzed directly. Capture-HPC, a client honeypot system, was used to perform the analysis. The client honeypot was installed in the virtual machine, which browses webpages by simulating the environment of a Windows client. All events triggered during the browsing were recorded, including the I/O operations of the file system, changes in the registry, and creation and destruction of processes. Finally, the normal events that were triggered by the operating system or the browser were excluded, and the recorded documents were sent to the behavior analysis module for assessment.

Table 5. Abnormal events in the file system exclusion list

Access	File Path	Access	File Path
Write	.+\.bat	Write	.+\.scr
Write	.+\.cmd	Write	.+\.wsc
Write	.+\.exe	Write	.+\.wsf
Write	.+\.inf	Write	.+\.wsh
Write	.+\.lnk	Write	.+\.vb
Write	.+\.msi	Write	.+\.com
Write	.+\.msp	Write	.+\.pdf
Write	.+\.pif	Write	C:\\WINDOWS\\win.ini
Write	.+\.reg	Write	C:\\WINDOWS\\Tasks\\.+
Write	.+\.sct	Write	C:\\Documents and Settings\\.+ \\Start Menu\\Programs\\Startup.+
Write	.+\.shs		

Exclusion lists were set for the file system, registry, and processes. Besides the normal events that were excluded, the abnormal events that should not be excluded were reserved.

Previous studies [2] have demonstrated that although dynamic analysis does not yield false positives, it produces false negatives because a false negative can occur if no malicious behavior occurs at the time of detection. This study uses an exclusion list and conditional constraints to reiterate the detection to ensure false negatives are avoided. Table 6 shows the results of purely dynamic analysis, with an accuracy rate of 100%. The total runtime was approximately 27 h and 55 min.

Table 6. Decisions for dynamic analysis only detection

Result	Malicious 477 entries	Benign 598 entries
Malicious	477	0
Benign	0	598

V. CONCLUSIONS

This study proposed a mechanism for malicious web detection. According to the data collected in the experiments, static analysis only has relatively low accuracy and dynamic analysis is time-and resource consuming.

Blind spots still exist in the identification of malicious websites. Currently, static analysis relies primarily on keywords to perform detection; however, these keywords may be altered afterward, preventing the system from identifying them correctly. Furthermore, after the current static analysis of this study, most benign webpages were regarded as suspicious and their behavior malicious; thus, the detection time was subsequently prolonged. The techniques for detecting benign webpages maybe improved in the future to reduce the detection time.

ACKNOWLEDGMENT

This research is partially supported by Grant No. NSC 101-2221-E-224-062 from the National Science Council of the Republic of China.

REFERENCES

[1] Alexa Top 500 Global Sites. [cited 2013/4/20]; Available from: http://www.alexa.com/topsites.

[2] Christodorescu, M., et al., Semantics-Aware Malware Detection, in Proceedings of the 2005 IEEE Symposium on Security and Privacy. 2005, IEEE Computer Society. p. 32-46.

[3] Christodorescu, M., et al., A semantics-based approach to malware detection. ACM Trans. Program. Lang. Syst., 2008. 30(5): p. 1-54.

[4] Capture-BAT. [cited 2013/4/20]; Available from: https://www.honeynet.org/node/315.

[5] Capture-HPC. [cited 2013/4/20]; Available from: https://projects.honeynet.org/capture-hpc.

[6] Google. [cited 2013/4/20]; Available from: http://www.google.com.

[7] Honeyclient Project [cited 2013/4/20]; Available from: http://www.honeyclient.org/trac.

[8] Honeyd. [cited 2013/4/20]; Available from: http://www.honeyd.org/.

[9] HoneyMonkey [cited 2013/4/20]; Available from: http://research.microsoft.com/en-us/um/redmond/projects/strider/honeymonkey/.

[10] IBM, 2009 Mid-Year Trend and Risk Report. 2009.

[11] Malware Block List. [cited 2013/4/20]; Available from: http://www.malware.com.br/.

[12] Malware Domain List. [cited 2013/4/20]; Available from: http://www.malwaredomainlist.com/.

[13] Seifert, C., et al., Drive-by-downloads. 2008.

[14] Seifert, C., et al., Capture - A behavioral analysis tool for applications and documents. Digital Investigation, 2007. 4(Supplement 1): p. 23-30.

[15] 18. Seifert, C., I. Welch, and P. Komisarczuk. Identification of Malicious Web Pages with Static Heuristics. in Telecommunication Networks and Applications Conference, 2008. ATNAC 2008. Australasian. 2008.

[16] Symantec, Web Based Attacks. 2009.

[17] Wikipedia. Drive-by download. [cited 2013/4/20]; Available from: http://en.wikipedia.org/wiki/Drive-by_download.