

Malicious Web Page Detection Based on Anomaly Semantics

D. J. Guan¹, Chia-Mei Chen², Jing-Siang Luo¹, and Yung-Tsung Hou²

¹ Department of Computer Science and Engineering

² Department of Information Management
National Sun Yat-Sen University, Kaohsiung, Taiwan

Abstract. Web services are becoming the dominant way to provide access to on-line information. Web services have a lot of applications, like e-mail, web search engine, auction network and internet banking. On the web services, web application technology and dynamic webpage technology are very important, but hackers take advantage of web application vulnerabilities and dynamic webpage technology to inject malicious codes into webpages. However, a part of the web sites have neglected the issue of security. In this paper, we propose a novel approach for detecting malicious webpages by URL features, anomaly semantics, potential dangerous tags and tag attributes. This research proposed approach mainly consists of three parts: (1) scripting language and automatic link filter. (2) malicious feature. (3) scoring mechanism. By first part, this step can filter out normal webpages to increase detection speed. Second part can identify some known malicious attacks. Third part can search some unknown malicious webpages by scoring. Our experimental results show that the proposed approach achieves low false positive rate and low false negative rate.

1 Introduction

World Wide Web (WWW) is perhaps the most significant technology of modern computing. World Wide Web allow us to search information, use online banking, go online shopping, use discussion on forum, receive news, be entertained, and communicate with others like Instant messaging (IM), at our convenience twentyfour hours a day and around the world.

Above-mentioned services, is the emerging technology called "web application". These technology change the web from a publishing platform to a processing platform. Enabled by web application, the web is not just about moving information; it's about moving information toward a purpose. For example, a seller posts inventory and price on its web site. This is only moving information. Now suppose that, at a consumer's request, a web service search the seller's site for a specific thing and the price threshold. When the thing appears in inventory and the below consumer's price threshold, the web service initiates the purchase, accesses the consumer's credit card and how to send thing, and finish all work. This is moving information toward a purpose.

Web application makes our life very easy. However, web application has in the risk of shallow. Recently, hackers used a mass of Structured Query Language (SQL) injection attacks involving over 10,000 webpages that had been injected with malicious JavaScript. A web browser may get infected and lose the sensitive data by browsing any one of these pages.

And sites had been injected with malicious code, phishing or other malicious websites, the user is difficult to see from the webpages with the normal differences. Some sites have also been injected by some deformation, encrypted or encoded code mechanism, such as obfuscation, even more difficult to identify users.

In recent years, the rise and develop of web applications, many web services are use scripting language. And network threats, browser vulnerabilities, let malicious programs increased. According to Symantec ISTR study, the growth of malicious programs in 2007 was 5.68 times in 2006, is 2008 times that in 2007 more than 2[1]. And the purpose of the hacker attacks on vulnerabilities from vulnerability to attack systems, to find methods to break the software, use computer viruses or worms to undermine attacks turned the target of the malicious programs, the more tangible benefits in order to obtain or make money for the purpose, such as the resale of the user's confidential information, manipulate the user's computer intrusion crimes and so on.

Faced with a huge data and information, complex and changing deformation mechanisms, obfuscation, as well as to attack the ever-changing tactics, making the market the sale of anti-virus software is facing significant challenges. Mihai Christodorescu and Somesh Jha [2] in 2004, was a commercial anti-virus software for a malicious software technology used by the obfuscation of tolerance to test. The results showed that various anti-virus software false negative rate as the average from forty percent to eighty percent, and even appears completely misjudging the situation, figure 1. And really identify whether the code or malicious website, be approved by the well-trained experienced security staff to manually make determination. In order to be effective in reducing the cost of security work, this paper attempts to use original URL and source code detection. We hope automatically to filter out the normal webpages and sort out the malicious website.

The rest of this paper is structured as follow: Section 2 discuss related work. Section 3 describe the detail of the proposed approach. Experimental evaluations are presented in section 4. Section 5 concludes this paper.

2 Related Work

Previous work about XSS attack, [3] presented client-side solution to mitigate cross-site scripting attacks. They establish a tools, Noxes. Noxes plug-in in the Microsoft personal firewall applicationsacts and acts as a web proxy and uses both manual and automatically generated rules to mitigate possible cross-site scripting attempts. The advantage is able to handle just using url and domain name. This paper is only to XSS attacks, it seems that for other types of attacks

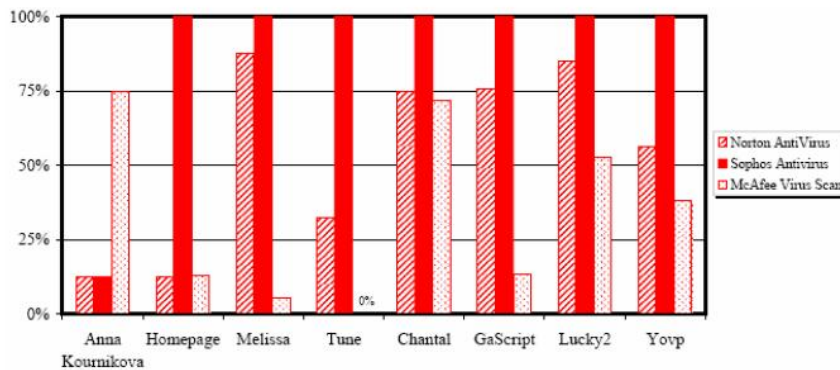


Fig. 1. The anti-virus software through the obfuscation of the malicious programs after deformation, the average rate of false negative rate (Source: [2]).

can not be processed. And because using white-list, the system needs to store large amounts of data.

And [4] presented static heuristics method to classify malicious pages. The author is using the machine learning to build a decision tree to distinguish a malicious webpage. And use high-interaction client honeypot Capture-HPC v2.1 to collect malicious samples. Finally, the false positive rate of 5.88 percent, 46.15 percent rate of false negative. The reason that feature is not enough, and decision tree methods to simple lead to high rates of false negative.

Lin [6] proposed model based reasoning and template mechanism, through signature and semantic to classify malicious DHTML. The method compared with signature-based mechanism, has a nice detection performance for polymorphism and metamorphism mechanism attempt to avoid detection. However, the algorithm also has its limitations, if a model has not been considered in the template, will have an error in the results generated, so the need for new semantic or feature do update and modify.

3 Proposed Approach

The purpose of this study is to detect malicious dynamic webpages. The detection of dynamic malicious webpages and general malicious programs (e.g., viruses, worms, backdoor programs, key logger) of major difference is that the former can directly browse the source codes, but the latter can not browse the source codes. Our detection method is the use of static analysis to determine page-source whether contains any malicious semantics or abnormal behavior.

The system detects the following processes. First of all, crawl all of the webpage links. The second step return encode, split and type conversion function. The third step detect the webpages whether the source codes invoke the scripting languageis and use to automatically link. If the two are not, this webpage

regards as normal page. The fourth step use the URL (Uniform Resource Locator) features, Whols information of website domain name, node relationship. Because these features do not view the page source, the system can effectively speed up the detection. And then, use of detection webpage source technology and malicious feature. The fifth step is the use of scoring mechanism. The system established threshold to predict whether the webpages are malicious page. Figure 2 illustrates the detection flow of our approach.

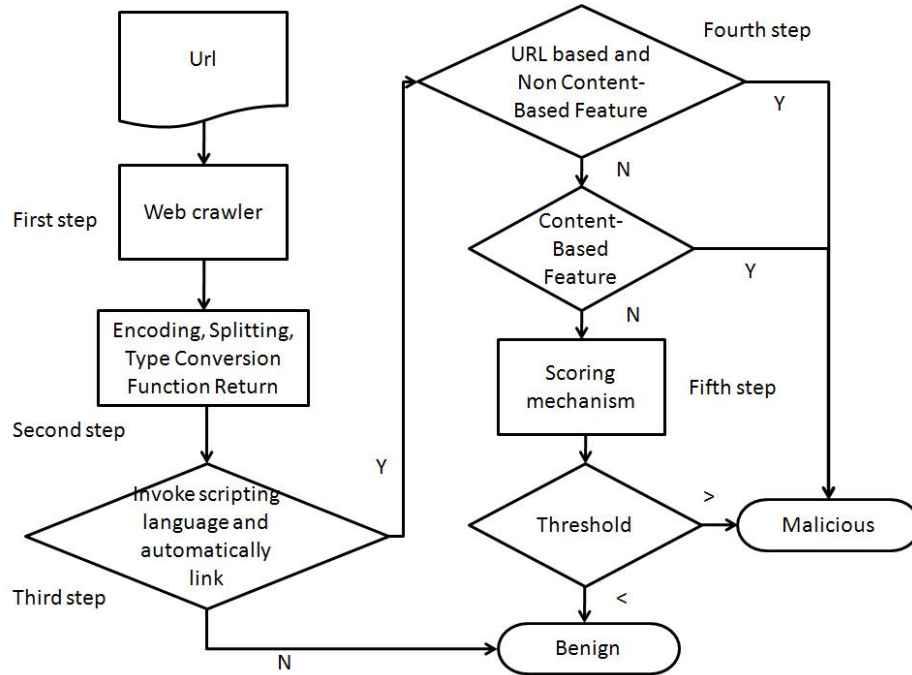


Fig. 2. The detection flow.

3.1 Web Crawler

Web crawling is to collect all kinds of links. To put it simply, the web crawler is an automated information collection and classification of network computer program. Web crawler also can be used automatic browse webpage. Many search engines website use the latest information are provided by web crawler. Crawler also can do maintenance work, such as checking links, HTML code inspection, etc.

Web crawler has a list of URLs to visit at the beginning, called the seeds. Web crawler will visit these URLs, and identify all the hyperlinks in the page. Finally, all the hyperlinks were added to the list of URL to be the next visit. This

act calls Frontier [7]. Come out of the URL you want to detect, web crawling action, and to take out the link on this page. Based on these links will continue to crawl, in accordance with the relationship of these links on behalf of the parent node and child nodes. This step records the relationship between parent and child nodes (e.g., including each node's father, each node's sons), records the number of node at each level, and records each node's level.

3.2 Encoding, Splitting, Type Conversion Function Return

In order to avoid detection, some hackers use of obfuscation technologies to prevent detection by antivirus software. This step hope to return to the obfuscation mechanism. Encoding is one of obfuscation. There are unicode, hexadecimal, binary, decimal, or encode for special browsers in webpage source codes. And then, splitting means one word change into more than one words, such as <iframe> become '<if' and 'rame>'. Finally, type conversion function is like `string.fromCharCode`, the function can change ASCII code into char. For example,

```
document.write('<'+string.fromCharCode(105)+'fram'+
unescape('%65')+'>');
```

can be returned as

```
document.write(<iframe>);
```

3.3 Invoking Scripting Language and Having automatically Link Tags

In order to increase the detection speed, this step can use invoke scripting language and automatically link tags information to filter out some of the normal page. If both are not, this page can regard as normal.

Invoking scripting language refers to the use of Scripting language (e.g., Java Script and VBScript and JScript). The following example illustrates how scripting language can be invoked:

```
<script language="javascript">alert('hello');</script>
```

```
<iframe src="javascript:alert('hello');">
```

```
<style type="text/javascript">alert('hello');</style>
```

And Js file.

Automatic link refers to browser will be automatically linked into the other links after compilation. Several HTML tags have automatically link characteristic, such as IFRAME, META, IMG, EMBED, XML, STYLE, OBJECT, APPLET tags. The following example illustrates how META and STYLE tags can automatically link:

```
<meta http-equiv="refresh" content="0; url=http://badlink">
```

```
<link rel="stylesheet" href="http://badlink">
```

After this step, the system can filter out about 10% normal samples.

3.4 URL based and Non Content-Based Feature

This step defined some URL based and non content-based feature to increase detection speed. There are suspicious URL feature, fresh domain and node relationship. These feature are based on our observations of benign and malicious samples.

- **Scripting Language Behind in The URL:** In normal link, URL will only contain the domain name and file path name. If the link appears script tag or script code, the link can be regarded as suspicious. For instance,

```
<a href="http://example.com/comment.cgi?mycomment=<script>
maliciouscode</script>">Click here</a>
```

```
<a href="http://example.com/comment.cgi?mycomment=<script
src='http://badsite/badfile'></script>">Click here</a>
```

First will use script tag to inject the malicious code, and second will inject bad link.

- **URL Redirect to Different Domain Page Without Warning:** Redirection without warning such as redirection time set to 0, without any prompts, or no OK button to click. If redirect to the new site with the original website belong to different domain, on behalf of this website is not in accordance with the user actions. The webpage can be regarded as malicious. For instance,

```
http://g.msn.com.cn/CNCONH6/20806.11?http://bad site

setTimeout("location.href = 'bad site'", redirection time);

location.href = ('bad site');

self.parent.location = "bad site";

window.location = 'bad site';

<meta http-equiv = "refresh" content="0;url=bad site">
```

- **File Name Do Not Match with Content:** Normal condition, html pages are written by the html tags, js files are written in Java Script language. Therefore, the content of html pages does not have html tags, or source code of Js files contains only a link, the page can be regarded as suspicious.
- **URL Download file without warning:** Normal page when the file download, the webpages may ask you that do you wish to download this file, or have attached a click for download, or display download window. And here the download without warning, refers to the user do not know to download an executable file. If there have the feature of webpages can be viewed as suspicious. For instance,

```
<iframe style="position: absolute; top: 10; left: 124;
width:546px;height: 524px; visibility: hidden" frameborder="0"
scrolling= "no" src ="http://94.247.2.157/.dif/go.php?sid=1">
</iframe>
```

- **Crawled a Binary File:** The webpages file size are more than the average twice, or greater than the threshold. And store of the page source in the text file is binary form. On behalf of this website may contain drive-by download mechanisms, will be downloaded to a binary file.
- **URL is Encoded:** Some hackers will be encoded links to make page-source difficult to know where the link is directed to. Such as the following example, example 1 is a Hex encoding, and Example 2 is the percent encoding. Example 1 and 2:

```
\x2e\x2f\x69\x36\x34\x2e\x73\x77\x66
http://%77%2E%65%35%78%38%2E%63%6F%6D/js.js
```

after reduction will be

```
./i64.swf
http://w.e5x8.com/js.js
```

Non Content-Based Feature:

- **WhoIs Registration Time:** Many hackers on the network, will apply for a large number of sites, and then take advantage of these new website as a phishing site to lure other users to enter sensitive information, or written into a malicious website to embedded into vulnerable site, and then wait for the user into the victim website. According to analysis report of [10] from the APWG, the average survival time were 49.5 hours, and median uptimes were 19.5 hours. So the threshold can set as two-days, then use this as a standard threshold to determine whether the suspicious page.
- **Node relationship of Eash Page:** Before recorded in accordance with the relationship between nodes, this information can be regarded as the feature. If a link is embedded in a malicious link or source, then the original malicious code or links extended the tree, which marked suspicious webpage. And then the system tests the source codes of this tree.

3.5 Content-Based Feature

- **Setinterval Function More Than ClearInterval:** Its function written as follows:

```
Setinterval("function()",1000);
```

This means that call function once for every second. Setinterval function was originally used when the swf file to play regularly function call, intention is to only allow the play head into the frame to use interval function, and

reduce the screen refresh each time the impact. The `clearinterval` function is used to stop the `setinterval` function. However, hackers use this function in inappropriate places, such as open prompt from time to time, open Ad from time to time, or execute malicious process.

- **Invisible Page Elements:** According to [6], this feature refers to some source codes linked to other webpage but user didn't know. For instance,

```
<iframe src="http://badsite" width=0 height=0></iframe>
```

where `width=0` and `height=0` are Invisible.

- **Js File Only Line of Code:** Js file originally was intended for a number of programmers commonly used Java Script function in each webpage can be used, do not have to rewrite function of each webpage. Including one line code will contain all of the function. For example as follows:

```
<script src="js/pngfix.js" type=text/javascript></script>
```

So, if the Js file only has line of code, it's not in conformity with the intention of Js files. The webpage could be regarded as suspicious feature.

- **CSS Attack:** CSS is an acronym for Cascading Style Sheets. The main purpose of CSS is trying to separation of document structure (using HTML or other languages written in) and document display, which can more readable and flexible. However, some hackers will use websites which allows the user to modify their own CSS to inject malicious CSS. Therefore, if found to modify the CSS code has link outside the website domain, then the webpage can be viewed as suspicious.
- **Arp Attack:** If there is a malicious server in the LAN, server will add some malicious code when you send HTTP request. This called "arp attack". In our malicious samples, there are some arp attack source code using VBScript language.

3.6 Potential dangerous of tags and tag attributes

Then according to [5,9] use potential dangerous tag and tag attribute as the feature of malicious weight for each page. If there are a certain tag and tag attribute features, as a suspicious webpage, given the right kinds of weight.

- **SCRIPT Tag:** After use, page source code may be embedded in script code (such as Java Script, JScript, VB Script). However, this tag may be misused. For example,

```
<script src ="http://hacker/js.js" type=text/javascript>
</script>
```

where the link in script tag is designed by hackers.

- **APPLET Tag:** After use, will be able to embed small programs, which are prepared by a variety of programming languages, the browser can automatically download and implement. However, this tag may be inappropriate use, such as hackers embed malicious programs into the webpages, will harm to user.

- **EMBED Tag:** Using this tag could be inserted into the voice file, music files or images file to the webpage. However, the hacker inserted fake music files or image files, this tag becomes dangerous. For example,

```
<embed src="http://hacker/bad.swf" width="560" height="560">
</embed>
```

where the link is fake.

- **OBJECT Tag:** To use this tag, music files (e.g., ra, wav) and image files (e.g., avi, mov) can be set. This function is similar to the EMBED tag, it may have been misused. For instance,

```
<object data="javascript:bad code;"></object>
```

- **XML Tag:** This tag is the use of digital transmission and to carry information, its function is more extended than the HTML tag, xml tag can be self-describes. However, this tag may be misused. For instance,

```
<xml id ="abc"><a><b><script>badcode</script>;
</b></a></xml>
```

- **STYLE Tag:** In order to use CSS, need style tag. The CSS can define the attributes, background colors, text, and border-related. However, if misused, will harm to user. For instance,

```
<style type="text/css">
Body{
    background-image:url(javascript:
    open('http://www.X.com/muma.htm'))
}
</style>
```

This example is set to the background pattern, the injection of javascript to the site designated by hackers.

- **FORM Tag:** When the page needs to be entered in the information back to the web server, the webpage will be used in FORM tags. One can join a number of control that allows users to import information more convenient. However, this tag will be used by hackers.
- **META Tag:** This tag used to define itself on the information page for the web server or browser. There are very helpful, such as the author information, describe webpage content and set the keyword. Examples of the misuse,

```
<meta http-equiv="refresh" content="0;url=
javascript:bad code">
```

- **IFRAME and FRAME Tag:** In order to make classification and page management information more easily, the webpage need to use IFRAME and FRAME tags to the screen is separated into different frameworks, each framework are contained in a separate HTML document. The framework can also be inter-connected display. However, this tags is often used by hackers. For example,

```
<iframe src="http://badsite" width=0 height=0></iframe>
```

- **IMG Tag:** This tag is used to show the picture. Webpage designers must be specified src attribute of the URL where the graphics file. However, if an attacker insert fake image file, this tag will be dangerous. For instance,

```

```

- **SRC Tag Attribute:** This tag attribute is used to link picture. However, hackers may be misused this attribute. For example,

```

```

3.7 Scoring Mechanism

When a webpage source code does not match one of the malicious patterns, the system must calculate the score using tags and tag attributes in section 3.6. The scoring mechanism we developed depends on a set of training data, which include benign data and malicious data.

This step select 7 tags and 1 tag attribute to estimate the score of a webpage, and use potential dangerous tags in each of the pages of the proportion as webpage weight. This mechanism define $Weight(P_i)$ as the weight of i th webpage, where P_i is the proportion of each potential dangerous tag and attribute. The scoring strategies of tag and tag attribute are summarized in Table 1.

Table 1. Tags and attributes calculation

Tags and attributes	Scoring
Tag1: SCRIPT Tag	$Weight_1(P_i)$
Tag2: IFRAME Tag	$Weight_2(P_i)$
Tag3: APPLET Tag	$Weight_3(P_i)$
Tag4: EMBED Tag	$Weight_4(P_i)$
Tag5: OBJECT Tag	$Weight_5(P_i)$
Tag6: STYLE Tag	$Weight_6(P_i)$
Tag7: META Tag	$Weight_7(P_i)$
Attribute1: SRC Tag Attribute	$Weight_8(P_i)$

We can say that sum of these weight is webpage proportion of potential dangerous tags besides SRC tag attribute. The following equation are proportion of potential dangerous tags, where $Proportion_{pdt}$ means the proportion of potential dangerous tags.

$$Proportion_{pdt} = \sum_{j=1}^7 Weight_j(P_i)$$

4 Experimental Evaluation

In this section, we evaluate the performance of our proposed system. And experimental data have two sets of webpages, malicious and benign sample. Malicious webpages sample are collected from <http://www.itis.tw/badlink>, <http://rogerspeaking.com>, <http://tool.ikaka.com>, network administrator of an anonymous hospital and obeyed the privacy principle, and some are from Google. According to [8], over 50% of malicious websites are located in China. Therefore, we use Google search by keywords facilitates for getting malicious webpages. As for benign webpages collection, we develop a crawler to collect webpage source codes from Google, Yahoo and MSN. In total, we have 571 benign webpages and 259 malicious webpages in our experiment.

Then we use four metrics to measure the performance of our approach:

$$\begin{aligned} \text{False Positive Rate (FP)} &= \frac{N_{ben-mal}}{N_{ben}}, \\ \text{False Negative Rate (FN)} &= \frac{N_{mal-ben}}{N_{mal}}, \\ \text{Accuracy} &= \frac{N_{cor}}{N_{tot}}, \\ \text{Detection Rate} &= \frac{N_{mal-mal}}{N_{mal}}, \end{aligned}$$

where N_{ben} is the number of benign webpages; $N_{ben-mal}$ is the number of benign webpages that are misclassified as malicious webpages; N_{mal} is the number of malicious webpages; $N_{mal-ben}$ is the number of malicious webpages that are misclassified as benign webpages; N_{tot} is the number of benign webpages and malicious webpages; N_{cor} is the number of correctly classified; $N_{mal-mal}$ is the number of malicious webpages that are classified as malicious webpages.

Table 2 shows the experimental data:

Table 2. Experiment results

False Positive Rate	23/571 = 4.028%
False Negative Rate	1/259 = 0.386%
Accuracy	806/830 = 97.108%
Detection Rate	258/259 = 99.614%

5 Conclusion

In this paper, we present an anomaly semantics approach for detecting malicious webpages. This approach is based on URL features, non content-based features, content-based features, potential dangerous tags and tag attributes. We define 13 malicious features to detect webpage attack. Also using a scoring mechanism by 7 potential dangerous tags and 1 tag attributes to predict a webpage that

does not match any malicious feature. In our experimental results, which show that our approach achieves low false positives and low false negatives.

In the future, we will collect more malicious samples to make our experiments more convincing and accurate. Moreover, the system will be embedded in the browser on real-time detection of malicious webpages.

References

1. iThome Symantec: malware expert is expected to exceed the deformation, <http://www.ithome.com.tw/itadm/article.php?c=53448>
2. M. Christodorescu, and S. Jha.: Testing malware detectors. In: Proceedings of the ACM SIGSOFT International Symposium on Software Testing and Analysis. (2004)
3. Engin Kirda, Christopher Kruegel, Giovanni Vigna, and Nenad Jovanovic.: Noxes: A Client-Side Solution for Mitigating Cross-Site Scripting Attacks. In: SAC. 23–27 (2006)
4. Christian Seifert, Ian Welch, Peter Komisarczuk.: Identification of Malicious Web Pages with Static Heuristics. In: Telecommunication Networks and Applications Conference (2008)
5. Filtering JavaScript to Prevent Cross-Site Scripting. In: EUROSEC (2005)
6. Shih-Fen Lin.: Malicious DHTML Detection by Model-based Reasoning. (2007)
7. Web Crawler, http://en.wikipedia.org/wiki/Web_crawler
8. May 2008 badware websites report, http://www.stopbadware.org/pdfs/StopBadware_Infected_Sites_Report_062408.pdf
9. ha.ckers XSS (Cross Site Scripting) Cheat Sheet, <http://ha.ckers.org/xss.html>
10. APWG releases Global Phishing Survey: Domain Name Use and Trends in 1H2008, http://apwg.org/reports/APWG_GlobalPhishingSurvey1H2008.pdf