Hybrid Obfuscated Javascript Strength Analysis System for Detection of Malicious Websites

R. Krishnaveni, C. Chellappan, R. Dhanalakshmi

Hybrid Obfuscated Javascript Strength Analysis System for Detection of Malicious Websites

R Krishnaveni[1] , C. Chellappan [2], R.Dhanalakshmi [3]
Department of Computer Science & Engineering, Anna University, Chennai, India
{rskichu10@gmail.com[1], drcc@annauniv.edu[2], dhanalakshmisai@gmail.com[3] }

Abstract - JavaScripts are mostly used by the malicious websites to attack the client systems. To detect and prevent this, static and dynamic analysis systems are used which has problems like longer analysis time, setting up of virtual environment and prone to real attacks. Hence a new hybrid analysis system is proposed which reduces the shortcomings of the static and dynamic analysis systems. Additional features such as keywords to words ratio, average line length, presence of suspicious URLs and tags, whitespace percentage, number of redirections, and enigmatic variable names are used to measure the strength of the obfuscation. In this system performance is improved and the number of false positives and negatives are reduced. Based on the strength of obfuscation in the JavaScript code, a website is determined to be benign or malicious.

Keywords **–**Malicious Web Sites, JavaScript Obfuscation, JavaScript Extraction, Hybrid Strength Analysis System.

1 Introduction

Normally, the attacker uses the Internet to insert an attack code into the web site, or downloads the attacking program in the client's system without the user's knowledge. JavaScripts are the most convenient tools for the attackers to create a malicious web site. JavaScript obfuscation techniques are also used to hide the JavaScript containing the attack code.

Attack Sites are Web sites that try to infect our computer with malware when we visit. These attacks can be very difficult to detect; even a site that looks safe may be secretly trying to attack us. Sometimes the Web site's owner won't even know that the site has been turned into an Attack Site. Some malicious codes get distributed only under certain conditions; some installs silently, without alerting the user. There are three common ways for a website to become an attack site:

- The site is created for malicious purposes.
- The site serves malicious content from an ad network or other third-party content provider.
- The site is infected through a security weakness in the site or the Webmaster's computer.

A malicious website facilitates the distribution of malware, either intentionally or because it has been compromised. Most malicious websites distribute malware without the knowledge of the sites' owners.

1.1 JavaScript Obfuscation

Obfuscated JavaScript is a source or machine code that has been made difficult to understand for humans. Obfuscators transform readable code into obfuscated code using various techniques. There are mainly four methods in which JavaScript can be obfuscated.

- Using ASCII values and Unicode values
- Using an XOR operation
- Splitting a string
- Compressing a string and replacing with a meaningless string

2 Literature Survey

2.1 Static Analysis Systems

Static analysis system downloads and analyzes the source of the web site. This system has less vulnerability against the attack, as the attack is not directly made. The shortcomings are that the person should analyze the source code manually and takes a longer amount of time to detect a variant and generate a signature.

Malzilla,[1] proposed in "Rhino:Javascript for java" by Mozilla is the method of downloading the source codes by entering the URL of the web site and then analyze them. Until the normal JavaScript is obtained, decoding of the obfuscated codes takes place. There is a shortcoming in analyzing whether the detected JavaScript code is an attack JavaScript that induces malicious behavior or not.

The method proposed in "Automatic detection for JavaScript Obfuscation Attacks in web pages through string pattern analysis"[2] by YoungHanChoi analyzes the pattern of web page strings. It separates obfuscated JavaScript codes from normal ones using N-gram entropy and string size. If JavaScript is found to exceed the certain threshold value, the web site will be considered as a malicious web site. As the characteristic of the malicious code is not considered, it could be difficult to detect a malicious web site that contains normal JavaScript.

The method in "Access Log Generator for Analyzing Malicious Website Browsing Behaviors"[3] by Chu Hsing Lin proposed an access log generator to generate access logs with characteristics of browsing behaviors for the particular website under investigation. We also include malicious behaviors into the generated access log, which is combined with actual access log of the website for further tests and analyses. The disadvantage of this concept is that the website is not checked for malicious scripts at run time. The method is slow and prone to attack before the access logs are analyzed.

The approach in "Beyond Blacklists: Learning to Detect Malicious Web Sites from Suspicious URLs"[4] by Justin Ma for detection of malicious websites analyzes the lexical properties of the URL. The disadvantage of this method is that only the suspicious URL is analyzed to find whether a website is benign or malicious.

The anomaly semantics approach in “Malicious Web Page Detection Based on Anomaly Semantics”[5] by D.J.Guan for detecting malicious web pages is based on URL features, non content-based features, content-based features, potential dangerous tags and tag attributes. The disadvantage of this method is that it mainly concentrates on the HTML tags and attributes and does not concentrate more on JavaScript.

2.2 Dynamic Analysis Systems

The dynamic analysis system visits the web site to check if any of the web pages causes any change in the user’s system in order to analyze malicious behavior of the web site. The web site obfuscated with JavaScript is not directly decoded. Instead, JavaScript codes are executed, and then the result is analyzed to check malicious behavior.

In Wepawet proposed by Marco Cova in “Detection and Analysis of Drive-By-Download Attacks and Malicious Javascript Code”[6] visits the web site using the entered web site address from the virtual environment system, developed for analysis and the visit result is analyzed. The virtual environment system analyzes whether the visited web site is obfuscated, whether any malicious behavior is committed or not and shows the analysis result.

MonkeyWrench proposed by Armin Buscher in “Throwing a MonkeyWrench into Web Attack Plans”[7], analyzes the web page using an analysis method similar to Wepawet.

Therefore, malicious behavior can be analyzed only when that particular application has been developed in a virtual environment in advance. Hence, much time and efforts are required to build these analysis environments.

The dynamic analysis method is fast and shows more accurate examination results than the static analysis method. It has the advantage of analyzing JavaScript codes that are generated dynamically. The dynamic analysis method can check the malicious nature of the web site within shorter time than static analysis. However, an environment should be created that allows execution of malicious behavior contained in the malicious obfuscated web site.

2.3 Hybrid Analysis System

The method proposed in “Suspicious Malicious Web Site Detection with Strength Analysis of a JavaScript Obfuscation” by “Byung-Ik Kim”[8], detect malicious websites based on JavaScript obfuscation strength check system. They analyze using limited characteristic features such as density, frequency and entropy. Due to this, accuracy and detection rate might be less.

3 System Architecture

An URL of the website is given as an input to the system. The codes of the website are analyzed and the JavaScript codes are extracted separately. The extracted

JavaScript code is analyzed using the metrics such as presence of suspicious URLs, density, frequency, enigmatic variable names, quasi- tautology, number of redirections, empty code branches and presence of suspicious URLs and scored based on the metrics. Based on the scores got by each website, a website is determined to be benign or malicious.

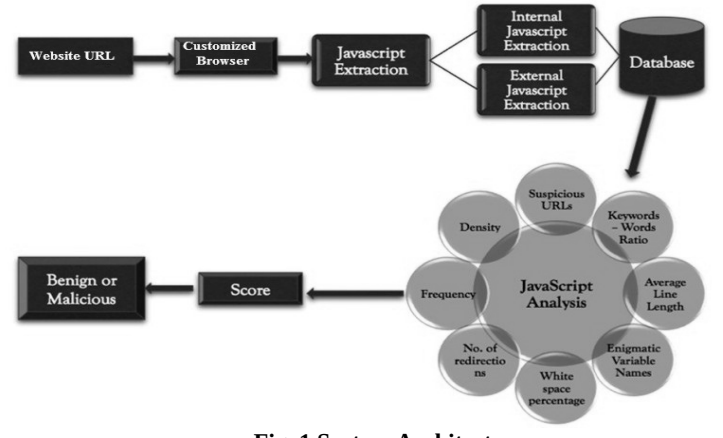

Fig. 1 System Architecture

4 Modules

The modules in the system are Custom Browser Creation, JavaScript Extraction, Obfuscated Javascript strength analysis and Scoring Mechanism

4.1 Custom Browser Creation

A Web browser is a software application used to locate, retrieve and also display content on the World Wide Web, including Web pages, images, video and other files. As a client/server model, the browser is the client run on a computer that contacts the Web server and requests information. The Web server sends the information back to the Web browser, which displays the results on the computer or other Internet-enabled device that supports a browser. A custom Web Browser is created to illustrate how this system can be integrated into the real time Web Browser.

4.2 JavaScript Extraction

The JavaScript present in the webpage is extracted from the other codes for analysis. This module extracts both internal javascript and external javascript. The internal javascript are extracted using the regular expressions by matching the patterns within the script tag. All the contents present inside the script tag are extracted and stored for analysis.

The external javascripts are scripts that are stored in a separate file and are referenced by the websites which uses it. The external javascript has the extension

'.js'. All reference to the '.js' file from the website are searched and the contents of the '.js' files are extracted and stored.

4.3 Obfuscated JavaScript Strength Analysis

The JavaScript that are thus extracted are analyzed using the following features to determine whether a website is malicious or benign.

1) Presence of suspicious URLs: All the URLs that are present in the website code are extracted separately and stored into the database. These URLs are compared with the already blacklisted URLs that are collected from various sources. The database contains around 60,000 blacklisted suspicious URLs, which are compared with.

$$\% \text{ of Suspicious URLs} = \frac{\text{No.of links matching with blacklist}}{\text{Total No. of links in the website}}$$

2) Density: Generally, the obfuscated string is longer than the normal string. The length of a string tends to increase while encoding a string or replacing it with a meaningless string. Therefore, this feature checks if the length of a single string is longer than 200 characters, if so, the website is considered to be malicious website.

$$\text{Density} = \frac{\text{Strings having more than 200 characters}}{\text{Total Number of Strings}}$$

3) Frequency: The JavaScript obfuscation strength check system uses the frequency of the particular function, encoding mark, and % symbol occurrence, as the detailed check items to check frequency. The proposed system checks the possibility of obfuscation, by checking the use frequency of these functions. Those that are considered dynamic functions are also checked.

$$\text{Frequency} = \frac{\text{Number of suspicious functions and symbols}}{\text{Total Number of Strings}}$$

4) Number of Redirections: The web page redirects the user to the different web pages automatically or through user's action such as clicking some ads, etc. The redirected web page may contain malicious code, which will be downloaded to the user's computer.

$$\text{Redirection Percentage} = \frac{\text{No. of redirections}}{\text{Total No. of Links}}$$

5) Presence of white spaces: Number of extra white spaces, line feeds and tabs are used in the code to confuse the strength analysis system.

$$\text{Whitespace Percentage} = \frac{\text{Number of whitespaces}}{\text{Total Number of Strings}}$$

6) Enigmatic Variable Names: The strings used in the javascript code may be meaningless when compared with the normal javascript code.

$$\text{Enigmatic Variable names} = \frac{\text{Number of meaningless variables}}{\text{Total Number of variables}}$$

7) Average Line Length: A normal javascript code might have a line length of less than 120. Traditional line length of javascript is 80. The obfuscated malicious javascript code might contain the lines which are greater than 120.

$$\text{Average Line Length} = \frac{\text{Number of lines having with length more than 120}}{\text{Total number of lines}}$$

8) Keywords to words ratio: In the malicious javascript code, the use of these keywords is limited due to large use of other operations such as multiple instantiations, declarations, function calls, etc. So in a normal javascript code the keywords are used more than in a malicious javascript code.

$$\text{Keywords – words ratio} = \frac{\text{Total keywords in javascript}}{\text{Total Number of words}}$$

4.4. Scoring Mechanism

A scoring mechanism is developed using the features that are used to check the websites. Each feature is tested whether the website satisfies its criteria. Each feature is deemed to be positive based on the analysis and by using the following scores.

- Presence of suspicious URLs – If more than 10% of URLs that are present in the website matches with the blacklisted URLs in the database, then this criterion is positive.
- Length – All the length of the strings present in the javascript code are calculated and if more than 30% of strings in the code is having more than the length of 200, then this feature is positive.
- Frequency – The number of suspicious functions present in the javascript code is found and if more than 30% of the string in the code is suspicious function strings, then this feature is positive.

- Number of redirections – If the percentage of number of redirected links to the number of actual links present in the website is more than 30%, then this feature is positive.
- Presence of whitespaces – if the ratio of whitespace to strings in the code is more than 30%, then this feature is positive.
- Enigmatic Variable Name – if more than 30% of variable names are meaningless in the javascript, then feature is positive.
- Average line length – if more than 30% of lines in the website has length of more than120, then this feature is positive.
- Keywords to words ratio – if only less than 10% of strings in the website are keywords, then this feature is positive.

Based on these scores, as shown in Fig 2, if a website is tested to have more than 2 features positive, then it is deemed to be malicious. If even only one of these feature i.e. presence of suspicious URLs or frequency is positive, then this website is found to be malicious. Here, we consider a sample of 100 malicious websites, and based on the proposed scoring mechanism, we observe that if we consider 1 positive feature alone as threshold the false positive rate is high and if 3 or more positive features are considered as threshold the detection rate decreases, hence the performance is poor. Hence the threshold is chosen as shown in Fig 2 which gives a better performance when tested.

```
If(no.of positive criteria>=2||presence of suspicious
urls=true||frequency=true)
  then
    open the alert page and block the malicious website from the user.
  else
     Open the website directly to the user.
Endif
```

Fig. 2 Scoring Pseudo Code

5 Performance Analysis

5.1 Analysis

Accuracy is the ratio of the number of correctly classified websites to the number of total websites. Correctly classified websites are the benign websites detected as benign websites and malicious websites detected as malicious websites.

$$\text{Accuracy Rate} = \frac{N_{cor}}{N_{total}} = \frac{N_{Ben-Ben} + N_{Mal-Mal}}{N_{total}}$$

Detection Rate is the ratio of number of malicious websites detected as malicious websites to the number of total malicious websites and ratio of number benign websites detected as benign websites to the number of total benign websites.

$$\text{Detection Rate} = \frac{N_{Ben-Ben}}{N_{Ben}} + \frac{N_{Mal-Mal}}{N_{Mal}}$$

$N_{Mal-Mal}$ = Number of malicious websites detected as malicious websites

$N_{Ben-Ben}$ = Number of benign websites detected as benign Websites

N_{cor} = Number of correctly classified websites

N_{total} = Number of total websites taken for analysis

5.2 Result

The table 1 shows the results of analysis of the websites. A total of 30 websites were taken for analysis and the above results were obtained. The accuracy is improved by 14% and detection rate is improved by around 2%.

Table 1. Comparison of the proposed system with existing system

Performance parameter	**Existing System**	**Proposed System**
Accuracy	83%	97%
Detection Rate	96%	98%
False Negative Rate	3.84%	0%
False Positive Rate	12.13%	4.35%

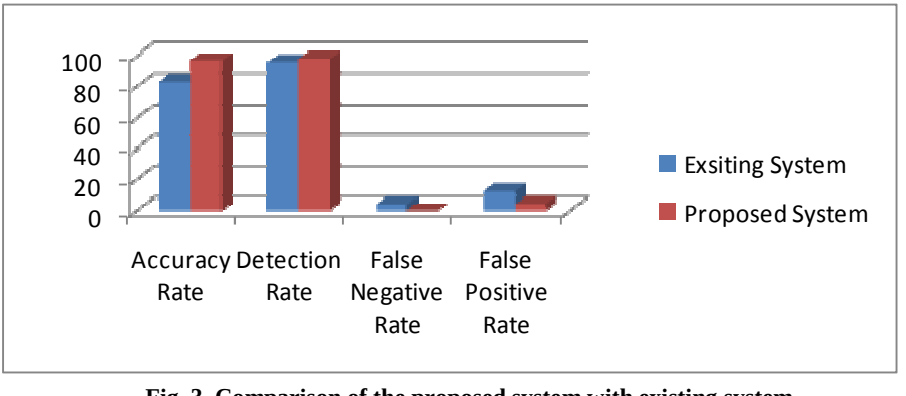

Fig. 3. Comparison of the proposed system with existing system

Fig. 3 shows the graph comparing the accuracy, detection, false negative, false positive rates of the existing system and the proposed system

6 Conclusion and Future Work

A new hybrid obfuscated java script strength analysis system was developed. The java script from the websites were extracted and analyzed for features such as presence of suspicious URLs, density, frequency, number of redirections, presence of whitespaces, average line length, enigmatic variable names and keywords to words ratio. A score was made based on these features which determined the website to be malicious or benign.

Analysis can be performed deeply to improve the scoring mechanism and processing speed which are considered as a future work. The system can also be extended to other scripts also.

References

1. Malzilla.org Rhino:JavaScript for Java. http://www.mozilla.org/rhino.
2. YoungHan Choi, TaeGhyoon Kim, SeokJin Choi, (2010) "Automatic Detection for Javascript Obfuscation Attacks in Web Pages through String Pattern Analysis", International Journal of Security and Its Applications, 4(2), pp.13-26.
3. Chu-Hsing Lin, Jung-Chun Liu, Ching-Ru Chen, (2009) “Access Log Generator for Analyzing Malicious Website Browsing Behaviors”, Fifth International Conference on Information Assurance and Security.
4. Justin Ma, Lawrence K. Saul, Stefan Savage, Geoffrey M. Voelker, (2009) “Beyond Blacklists: Learning to Detect Malicious Web Sites from Suspicious URLs”, KDD’09, Paris, France.
5. D. J. Guan, Chia-Mei Chen, Jing-Siang Luo, and Yung-Tsung Hou, (2009) “Malicious Web Page Detection Based on Anomaly Semantics”, fourth joint workshop on Information Security (JWIS 2009), Kaohsiung, Taiwan
6. Marco Cova, Christopher Kruegel, Giovanni Vigna, (2010) "Detection and Analsis of Drive-by-Download Attacks and Malicious JavaScript Code", Management of Computing and Information Systems.
7. Armin Bőscher, Michael Meier, Ralf Benzmőller, (2010) "Throwing a MonkeyWrench into Web Attacks Plans", International Fedration for Information Processing, pp.28-39.
8. Byung-Ik Kim, Chae-Tae Im, Hyun-chul Jung, (2011)” Suspicious Malicious Web Site Detection with Strength Analysis of a JavaScript Obfuscation”, International Journal of Advanced Science and Technology, Volume 26.