INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH & DEVELOPMENT

ISSN 2278 – 0211 (Online)

An Approach to Detect Malware Snippets

R. M. Yadav
Research Scholar, CSE Department, MANIT, Bhopal Madhya Pradesh, India
R. K. Baghel
Professor, CSE Department, MANIT, Bhopal Madhya Pradesh, India
Deepak Singh Tomar
Assistant Professor, CSE Department, MANIT, Bhopal Madhya Pradesh, India

Abstract:
With the proliferation in the internet technologies malware attacks also growing to steal credentials, money laundering, to take control on victim machines etc. Kaspersky give first rank to Malicious URL in top 20 Web based malwares. Attacker design web based malware snippets in such a way to automatically modify on the fly through obfuscation techniques. Most web-based applications are real-time by nature, this has the effect of significantly shrinking the timescale in which detection and enforcement decisions must be made. Due to these challenge, detection of web based malware through static analysis technique is a crucial task. This work introduced an algebraic based semantic static analysis technique to fast the detection process.

Keywords: *Malware, Metamorphic Malware, Polymorphic Malware, Web base Malware, Tokenization*

1. Introduction

The rising fame of Internet of Things such as web applications, portable devices, e-commerce social networking, etc builds the threat landscape a moving target for internet security. These systems could results in attacks that have a very personal impact on each of us. In addition, the web based malware is growing rapidly, which further increases risk. Kaspersky Laboratory [1] identifies that 93.01% of Web based attacks are the results of malicious URLs. According to the survey, Kaspersky placed malicious URLs at first rank in the list of top 20 most active Web based malware snippets.

Malwares are the malicious snippets intended to penetrate or take partial control of client's computer for the benefit of third-party. It is a collective term to refer a variety of hostile or intrusive snippets. Malware may controls the victim machine for the both malicious and benevolent purposes such as for advertisement, to track the activity, redirect client to attacker web, spread spams, launch DDoS etc. It may also invade the user privacy through accessing and modifying the personnel information.

Attacker may propagate malware through Shared Folders, Email Attachments, Fake Antivirus, Hijack browser, Fake Page, P2P Files etc. Also invent new malware invariants through obfuscation technique to bypass the detection mechanism. In this work first identify and gather the web based malware snippets from knowingly available databases. Second, carefully develop the advance versions of these malware snippets through code obfuscation techniques such as encryption, garbage code insertion etc. Advanced versions of web malware evade detection process to settle quickly on endpoints and spread throughout the network. Subsequently, an algebraic signature based web malware detection technique has been introduced to detect all developed versions.

2. Web Based Malware

Web-based malware are the malicious snippets distributed over the web to exploit the web vulnerability. It disseminated when victim visit malicious website. Web based malwares are becoming a prevalent threat in today's web environment. J Chang et Al. [2] identified millions of malicious URLs has been used as distribution point to spread malware snippets all over the Web. A malware snippet utilized to redirect the victim on attacker's web is as follows-

```
<iframe frameborder = "0" height= "0"
src="http://www.attackersite.com"
style="display:none" width="0"> </iframe>
```

In this snippet attacker fix value of height, width and border attributes of the <iframe> element to zero. This snippet redirects the victim on attacker's web site whose link is provided through 'src' attribute.

3. Malware Classification

K. Kaushal et Al. [3] classified malwares into two categories on the basis of code obfuscation techniques. Two categories of malwares are Polymorphic Malware and Metamorphic Malware.

3.1. Polymorphic Malware

It produces different variants of them while keeping the same functionality through polymorphic techniques. Polymorphic snippet is just the style of writing code that modifies the basic functionalities such as filename, compression, encryption etc. Main body of polymorphic malware consists of malicious code and encryption-decryption code. It eventually has to decrypt them before exploitation. Developing a polymorphic malware is complex process rather than the metamorphic malware. A polymorphic malware version of previous example is as follows-

```
<? Php
   eval(base64_decode ("PGlmcmFtZSBmcmFtZWJvcmR
        lcj0iMCIgaGVpZ2h0PSIwIiBzcmM9Imh0dHA6Ly9
        3d3cuZmxpcGNhcnQuY29tIiANCiAgc3R5bGU9Im
        Rpc3BsYXk6bm9uZSIgd2 lkdGg9IjAiPjwvaWZyYW
        1lPg0KPD9waHA="));
?>
```

This malware snippet uses two pre-defined PHP functions. First is *base64_decode ()* to decrypt code from base 64 and second is *eval()*to execute passed string as PHP scripts. Here, string passed into base64_decode () function is the base 64 conversion of malicious <iframe> code represented previously.

3.2. Metamorphic Malware

Metamorphic malware modifies their structure rather than its functionalities, each time it infects the system. New version utilizes obfuscation techniques such as Garbage Code Insertion, Register Usage Exchange and Subroutine Permutation to modify its structure. Metamorphic malware do not uses encryption technique to obfuscate the snippet as used in polymorphic malware. The main part of the metamorphic malware is Morphing Engine which obfuscates the snippets of the malware. A metamorphic malware version of same example is as follows-

```
  <iframe frameborder = "0" height= "0" name="attack"
 src="http://www.attackersite.com" width="0"
 style="display:none" scrolling="no" seamless > </iframe>
```

This malware snippet utilizes *style, name, scrolling* and *seamless* attributes in addition with *height, width* and *src* attributes. These extra attribute just changes the order of malware snippet rather than the functionality.

4. Challenges in Malware Detection

Web is a huge distributed system there is no single trust authority. Clients and web applications cannot trust blindly to each others. Typical web users may not have enough knowledge and expertise to protect them. Most web-based applications are real-time by nature which requires input from client side. These input fields are exploited by the attackers to penetrate the system and violate the privacy. Consequently blind trust on external data generates new vulnerabilities.

Web-based malware may easily leverage server-side polymorphism and spawn on the-fly. This has the consequence of producing large amount of web based malware on demand. It reduces the probability of capturing the sample and creating the signature. These challenges minimize the timescale of detecting and analyzing the web based malwares.

5. Background and Literature Review

This section presents the background review of static and dynamic malware detection techniques and illustrates the pros and cons of these techniques.

5.1. Static Analysis

In the field of static analysis, researchers focus on deriving the libraries of particular pattern on the basis of snippets or fixed behavior.

K. Jeong and H. Lee [4] proposed an algorithm using system-call graph to develop patterns and detect the malware snippets. In order to recognize metamorphic malware snippets, J. M. Borello et Al. [5] measure similarity between the behaviors of programs. Analyze the behavior of malware snippets through comparing the execution traces in terms of system calls. However, it suffers from the advanced code obfuscation techniques such as inserting the meaningless system calls. This obfuscation technique succeed to ignore the information of local functions, thus it may cause higher false positive.

Jusuk Lee et Al. [6] enhanced the system call graph technique to analyze malware snippets statically through classifying API calls in 128 clusters. Achieve the faster analysis through converting the API call sequence of malware into graph, known as call graph. Furthermore, K Kaushal et Al. [3] extracted the features of API calls in the unpacked executable binaries to detect executables files of metamorphic and polymorphic malware variants.

S. H. Shang et Al. [7] proposed an algorithm to determine the similarity between two malware snippets through function-call graph. This algorithm has been utilized to represent characteristics of malware snippets according to the similarity between function-call graphs. This technique is greatly utilized into detection and classification of different versions of malware snippets.
Ming Xu et Al. [8] propose a static analysis approach to identify the malware variants based on the function-call graph. Uses graph coloring technique to measure the similarity metric between two function-call graphs.
These techniques analyze the malware snippets without executing them. Static analysis is fast and safe rather than dynamic analysis techniques. It has difficulty to analyze the unknown malware snippets. However, existing malware detectors & antivirus programs usually utilize the static analysis techniques.

5.2. Dynamic Malware Analysis

This technique analyzes the malware through executing them. Detection technique first executes the malware into simulated environment, monitors the system calls and then observes its behavior. Usually virtual machines are employed as a sand box to execute the malware snippets and extract the behavioral features. TTAnalyze is a dynamic analyzer of windows executables. Dynamic analysis may detect unknown malwares. However dynamic analysis involves the risk of system infection.

5.3. Web Based Malware Analysis

Developing a safe browsing environment is the responsibility of the each web application security providers. Jian Chang [2] has done survey on different categories of web based malware detection techniques. At the last conclude that all techniques have its comparative pros and cons.
James Harland [9] discussed about web based malware snippets, infection lifecycle, and identified advanced code obfuscation and mutation techniques used to circumvent the detection of web based malware snippets. Niels Provos et Al. [10] analyze four propagation techniques to exploit of browser based vulnerabilities and presents example.
Manish Kumar Sahu et Al. [11] has done a review of malware detection techniques based on pattern matching. Finally conclude that the field of malware detection technique requires an optimized technique with less false positives. So that it is robust to detect byte or instruction level obfuscation through linear algebra.

6. Proposed Work

Proposed architecture to detect malicious snippets is based on linear algebra. It is divided into six sub modules which are Build Malware Repository, Semantic Analysis, Signature Formation, Algebraic Modeling, and Analysis of signatures. Block diagram of the proposed model is depicted into Figure 1.

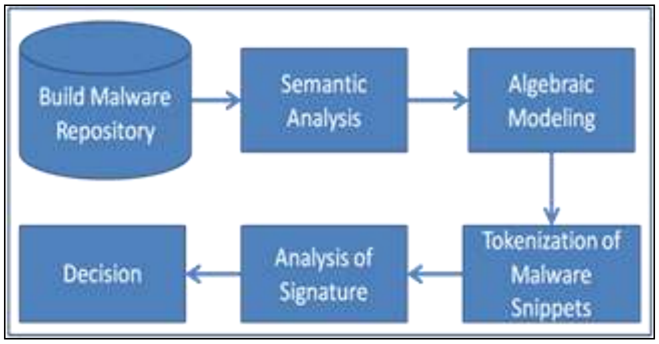

Figure1: Proposed Malware Analysis Model

6.1. Build Malware Repository

Firstly identify different malware snippets and there types commonly prepared through obfuscation such as encryption, garbage code insertion, register usage exchange and subroutine permutation. Two web applications Deep End Research [13] and VX heaven [14] have been preferred to build the repository of malware snippets.

6.2. Semantic Analysis

This module decompile the malware repository to view the code statically and attempt to understand behaviour of malware snippet. The <*iframe*> element is used to load an independent document or application into web page. It supports different formal and global attributes such as *src, height, width, border, event handlers* etc. In which height, width and border are the optional attributes and have default values. However the zero value of these three parameters hides the <*iframe*> contents from display. Attacker uses these hidden <*ifarme*> to do work without concerning the client which indicates toward the malicious intent.

6.3. Tokenization of Malware Snippets

Tokenization is the process to split snippet into elements and keywords of the used language. In this module first tokenize the snippet and create a database with their values. Here '*token_get_all()'* predefined PHP function has been used to tokenize the malicious snippet. It parses the malicious snippet into PHP language tokens through the use of '*Zend'* engine's lexical scanner. This function returns an array of token identifiers. Each individual token identifier is either a single character (*i.e.: ;, ., >, !, etc...*), or a three element array containing the token index in element 0, the string content of the original token in element 1 and the line number in element 2.

6.4. Algebraic Modeling

Basic structure of code functionalities has been carefully chosen to convey the relevant information. Here three attributes of <*iframe*> elements width, height, and border have been taken into consideration. Therefore algebraic operation for the basic structure of <*iframe*> has modeled in terms of their attributes, is as follows-

$I_{frame} : A_w \times A_h \times A_b \longmapsto B_{val}$

Where-

A_w = Represents the width of <iframe> element

A_h = Represents the height of <iframe> element

A_b = Represents the border of <iframe> element

$$A_i = \begin{cases} Having\ zero\ value & A_i = 1 \\ Having\ value\ rather\ than\ zero & A_i = 0 \end{cases}$$

$B_{val} = \mathbb{R} \longmapsto \{0, 1\};$

$$B_{val} = \begin{cases} Event\ occured\ successfully & B_{val} = 1 \\ Event\ fails & B_{val} = 0 \end{cases}$$

Here A_i represents the attribute values; incase of <iframe> considered three attributes are A_w, A_h and A_b. The zero value of these attribute assigns 1 to respective A_i. Otherwise for any value A_i is assigned with 0.

6.5. Analysis

In this phase, statistical examination of developed algebraic signature has been carried out through applying the token values of malware snippets. This signature utilizes the common characteristics of all three versions available in this work. Attacker set three attributes height, width and border of <iframe> tag to zero for hiding the <*iframe*> contents from the client. Second version modifies the snippets into base 64 conversions. Third version includes extra attributes such as *name, scroll, seamless and style* just to mislead the detection process. However common functionality of the all three version remains same. In all three cases attacker set zero to each attribute which assign 1 to each respective variables ($A_w = A_h = A_b = 1$). All ones represent the presence of attack. Developed signature is based on the common functionality. Therefore it viable to detect all versions of web based malware snippets presented into this work.

7. Conclusion and Future Work

This work successfully identified the web based malware snippets and carefully designed three versions of it to semantically analyze obfuscation techniques. According to the semantic behavioral analysis a linear algebraic signature has been developed successfully. This signature is based on linear algebra to provide ease of static web based malware detection process. Signature is working effectively for metamorphic and polymorphic version of it.

This work focuses on web based malware primarily on malicious ULRs. Future scope is to develop browser based plug-in on the basis of linear algebraic signatures to detect malicious URLs and other types of web based malware snippets.

8. Acknowledgment

This work is partially supported by the Network Security Laboratory of CSE Department, MANIT-Bhopal, India. We would like to thank all the anonymous reviewers for their insightful feedback and suggestions on our research work.

9. References

1. C. Funk and M. Garnaeva, "Kaspersky Security Bulletin 2013", December 2013, [online]. Available: http://securelist.com/analysis/kaspersky-security-bulletin/58265/ kaspersky-security-bulletin-2013-overall-statistics-for-2013/, [Accessed: 02/11/2014].
2. J. Chang, "Analyzing and defending against web-based malware", ACM Computing Surveys (CSUR), Vol: 45, Issue: 4, pp. 1-35, August 2013.
3. K Kausal, P Swadas and N Prajapati, "Metamorphic malware detection using statistical analysis", Int. J. of Soft Computing and Engineering (IJSCE), Vol: 2, Issue:3, July 2012.
4. K Jeong and H. Lee, "Code graph for malware detection", In: Proc. of the Int.Conf. on Information Networking, IEEE, pp. 1–5, 2008.
5. J. M. Borello, L. Me, and E. Filiol, "Dynamic malware detection by similarity measures between behavioral profiles" Proc. of the 2011 Conf. on Network and Information Systems Security, IEEE, 2011.
6. J Lee, K Jeong and H Lee, "Detection Metamorphic Malware using code Graph", Proc. of the ACM Symposium on Applied Computing, 2010
7. S. H. Shang, et Al., "Detecting malware variants via function-call graph similarity", Proc. of the 5th Malicious and Unwanted Software, IEEE, pp. 113–120, 2010.
8. Ming Xu et Al., "A similarity metric method of obfuscated malware using function-call graph", J. of Computer Virol Hack Tech, Vol: Issue: 9, pp. 35–47, Springer, France, 2013.

9. J. Harland, “Web-based Malware, Browser based Malware Infection”, [Online]. Avialable: http://www.auscert.org.au/download.html?f=1047, [Accessed: 11/11/2014]
10. N. Provos et Al., “The Ghost In The Browser Analysis of Web-based Malware”, Proc. of the first conf. on First Workshop on Hot Topics in Understanding Botnets, pp. 4-4, USA, 2007.
11. M K Sahu et Al., “A Review of Malware Detection Based on Pattern Matching Technique”, Int. J. of Computer Science and Information Technologies (IJCSIT), Vol: 5, Issue: 1, pp. 944-947, 2014.
12. D R Sahu and D S Tomar “DNS Pharming through PHP Injection: Attack Scenario and Investigation”, Int J of Computer Network and Information Security.
13. Mila At, “Deep End Research”, may 2013, [online] Available: http://www.deependresearch.org/2013/04/library-of-malware-traffic-patterns.html, [Accessed:11/11/2014].
14. VX Heaven, “Computer Virus Collection”, [online] Available: http://vxheaven.org/vl.php, [Accessed: 14/11/2014].