Asian Journal of Information Technology 15 (13): 2274-2281, 2016
ISSN: 1682-3915

Providing an Optimized Innovative System to Improve Security of Server-Side Scripts

Milad Moradi Rad and Nasser Modiri
Department of Electrical Computer and IT, Islamic Azad University of Zanjan, Zanjan, Iran

Abstract: Today, with development of internet and network environments all around the world, the necessity of security to work in cyberspace is felt more than ever. Information disclosure can cause great and sometimes irreversible damages to an organization. Therefore, in order to prevent direct access to the codes and manipulation of data, various ways such as obfuscation, encryption, cloaking, etc. have been suggested. In this study, we intend to obtain an acceptable conclusion in order to improve security of server scripts by presenting an integrated approach of simple techniques. The proposed scheme in addition to making the code unreadable in order to fulfil the copyright act, provides a relative security of the code against the attackers. At the end in order to prove the scheme, we provide a test condition and compare the results from running with the other available schemes.

Key words: Security of server codes, copyright protection, preventing injection attacks, code obfuscation, Iran

INTRODUCTION

With technology development, the number of internet users has been significantly increased in recent years and with the increase in number of internet users internet interactions such as communication and sales have had a growing trend. To launch their business, the users need to put the applications on the shared or dedicated servers; lots of developed applications are important and access to their contents means the loss of all efforts made for them. This importance can include the high cost of scheme development to the access to organization's security information. The great popularity of PHP language as a host platform (Tatroe *et al*., 2013) has turned it into a language of choice for the developers and vulnerable to the malware (Cholakov, 2008). A study conducted in the US National Vulnerability Database on April 2013 showed that about 30% of all vulnerability reports were related to PHP (Coelho, 2015). Although, this figure may seem alarming, but it is important to note that most of these vulnerabilities are not about the language itself, but are the result of poor programming methods used by the PHP developers. Hackers in the penetrated systems send spam and do other illegal activities through hosting the fake websites and exploit the victim server with distributed denial of service attacks or serving as anonymous platforms (Landesman, 2007). Loss of manufacturers' copyright, manipulation of the application by the spoilers, visibility of the information important to the hosting servers and easier access of the hackers to the program code are some of the challenges arisen by direct access to the codes on the servers. However, advanced engineering of the code can even cause the malware show a different behaviour (Sharif *et al*., 2008). But many of the attacks are deactivated with simple approaches and input control. To prove the efficiency of the scheme, we need to implement manual and automatic analysis and calculate the complexity of overhead using the actual malware samples (Schrittwieser *et al*., 2014). Later in the study, first we will have a look at the previous works in field of code security and become acquainted with the presented methods. Then, the proposed scheme will be discussed with the details of its stages. Next, we will evaluate the run output using the RIPS security testing software and will compare the results with a variety of models available in the market. The main purpose of the comparison is to create a safe framework in which the security issues of the script can be discussed accurately and the defects are resolved.

Review of literature: The studies conducted on security of server codes against the malware, apply four types of approaches. The first approach is control of the inputs in each system. The second approach includes the conceptual models and behavioural pattern recognition and the third approach is the malware recognition and applying effective solutions and running in protected modes. To achieve an understanding of the malware

Corresponding Author: Milad Moradi Rad, Department of Electrical Computer and IT, Islamic Azad University of Zanjan, Zanjan, Iran

capabilities, this approach was also developed to a system preventing the malware detection which uses the destructive behavioural patterns (Bayer *et al*., 2009) extracted by the running malware in a control environment (Kolbitsch *et al*., 2009). In addition, dynamic analysis of malware (Egele *et al*., 2012) has become an important concept to help processing the large-scale malware samples. Binary mutated clustering for behaviour analysis is useful as the sample model of malware systemic calls (Lanzi *et al*., 2010), this method still needs the ambiguous binary reverse engineering in order to discover very important and decisive trends including the domain generation algorithms, ambiguous encryption methods or cryptographic keys. In this study, architecture of Trojan detection system is presented which is an anti-ambiguity scalable system (Lu *et al*., 2013). This system is composed of several controllers, each responsible for one area of the network which receives the suspicious programs from the host. The research has studied the historical patterns of vulnerabilities in order to predict the future vulnerabilities in the applications and has shown the damages which have been reduced or increased from 2009-2014 (Murtaza *et al*., 2016). Various researches have been carried out in field of preventing the injection attacks. The research is an automatic detection system which includes a script finder with a list of XXS attacks and scans different areas of the website to detect the possible damages from JavaScript injection (Gupta and Gupta, 2016). The study discusses the different ways to provide security for web applications by injecting SQL and XSS codes (Deepa and Thilagam, 2016). The damages are classified in software development cycle phase and good solutions are recommended to exclude them. The accurate prevention study of XSS dynamic attacks (Bisht and Venkatakrishnan, 2008) is in charge of input validation and identification of the texts on the server side and omission of each text on the input. The approach of the study has created a security solution in web programs designed for behaviour detection by creating a proper response for each HTTP action. The research to avoid code injection to web applications through the website is in charge of analysis and filtering of the exchanged information and evaluating the runtime of the web browsers (Garcia-Alfaro and Navarro-Arribas, 2007). This research introduces the policy-based run of the web browsers and implementation of security policies defined in the server section as the basis for the work, carried out with the purpose of protecting the client from XSS attacks. Another approach is code obfuscation which leads to confusion and ambiguity in the codes for example, it renames the functions and variables with long strings to make the code reading and reverse engineering difficult. It should be noted that there will be no change in the program by ambiguity of the codes and the commands will run well (Brunton and Nissenbaum, 2015). Mehrian *et al*. (2015, 2016) provide a structural health monitoring using optimizing algorithms based on flexibility matrix approach and combination of natural frequencies and mode shapes. The purpose of this obfuscation is more to hide the code from the attackers and enhance the security of the code in term of author's copyright.

MATERIALS AND METHODS

The proposed scheme: In this part, we propose a combined multi-section structure by looking at the previous schemes. Each section includes procedures leading to improvement of the received script in order to be able to receive an executable and safe code. Importance of the specific data such as the connection strings in software, service connection codes, configuration file and access passwords has led us to pay an especial attention to obfuscation and encryption of the texts. The proposed solution includes the simple encryption combined operations, obfuscation and compression. Although, performing all innovative operations such as the proposed scheme cannot provide a complete stability, but in general, it is assumed that it will raise the costs of attacks. Operations of the proposed security solution are in form of Fig. 1. The first few stages of the proposed scheme are the pre-processing stages in other words input and preparation for algorithm's operations. The following gives a brief description of each stage.

Pre-processing: To apply security measures, a series of operations called the pre-processing are required to be performed. These operations lead to improvement in the generated output.

To create a prohibited list of variables and functions for better control of obfuscation operations, it is required to define a list of overall variable and the functions in default form and provide the possibility of a new variable and function by the user. This list has been added as the prohibited list, namely a part of the code not involved in obfuscation procedure. To configure the settings to set up the system, it is required to give initial values to a series of the variables, to take some inputs to be recorded in the program in a recognizable form in order to be easily recoverable in the future processes. Before implementing the main process, the duplication operation of all files is done, then the files are processed one by one. Therefore, to avoid more copies, the files are put directly in the main

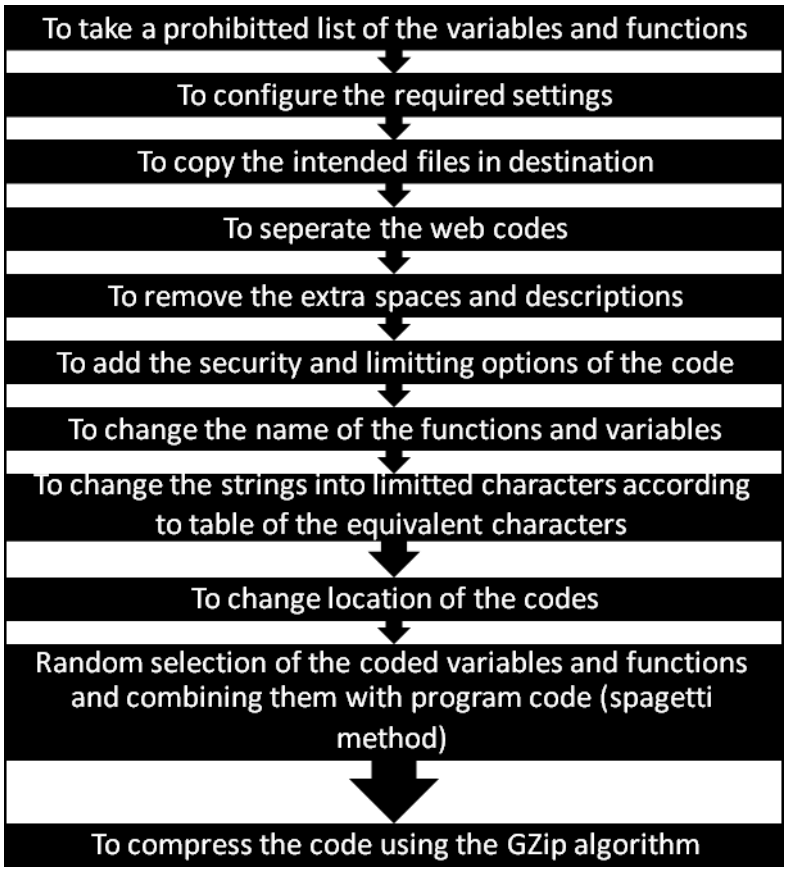

Fig. 1: Stages of the proposed algorithm

path instead of being transferred to a temporary path. Our Obfuscator algorithm is applicable on PHP codes. Therefore, there is a need to exclude the other languages such as js, css and html from obfuscation process; thus, we separate the PHP blocks or we can embed all other codes within PHP codes, to make the screen include the PHP codes uniformly but the approach of the proposed method is the way that unknown codes from the languages other than PHP will remain in the same way. Next stage is responsible for data cleaning duty. In this study, the voids, gaps and descriptions are deleted to reduce the code size and for faster processing of the operations. Descriptions in the run mode have no effect on the output and their presence cause decrease in running speed and increase the file size. During an automatic process of authorized blocks, distances and descriptions with different structural forms are omitted.

Security measures: Adding security and limiting options of the code is one of the capabilities of proposed scheme. Selections of the user on the code are required to be done in this stage. For instance, if there is a specified expiry date or copyright text, the restrictions are applied by user's selections. Possibility of choice by the user is optional and causes the inputs to be identified and appropriate filtering methods defined in the external class to be summoned before using the inputs. Finally, a security class file is created next to the obfuscated file. To obfuscate, names of the variables and functions must reach the lowest readability level. For example, a name between zero and one can be considered for the variables. The proposed scheme has used the MD5 function to mix the names of the variables and functions. Change of the strings into limited characters reduces the readability and strengthens the obfuscation process. We create a table like the ASCII table and add its equivalent strings. This table will be assessable by a function. The function is placed in the source codes or loaded in the server as a library file. Because of frequent use of the replacement process, the executive efficiency of the produced scripts is reduced, thus, the developer can select this location according to the conditions and interests (importance of the code and script low processing operation). Changing the location of the codes is another Obfuscation stage

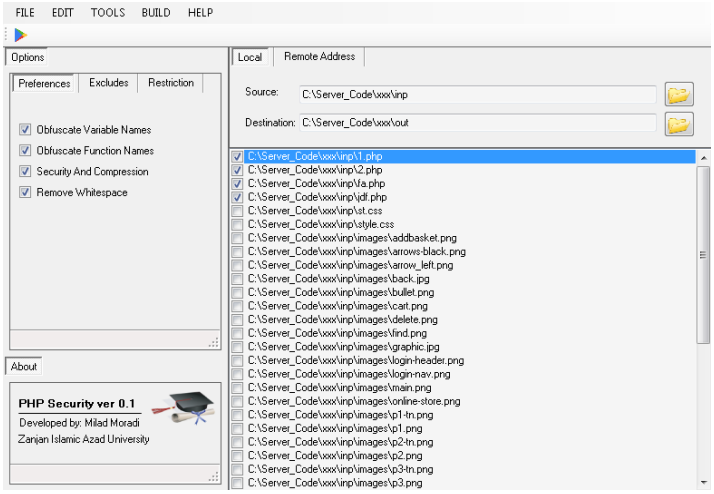

Fig. 2: PHP Security Software user interface (the proposed scheme)

where the functions and variables change place with no order. Combining the main codes with redundant codes can be an approach to mislead the code readers to avoid easy analysis of the program and reduce readability of the program. This capability in scheme implementation has been created by applying analysis in conceptual process of the code.

Post-processing: After the main obfuscation operations and putting the security methods in vulnerable sections of the code, we need to revive the output code. Volume of the output code may increase based on various actions of the user such as lack of code cleaning, obfuscation, changing location of the codes, selective change of the names (variables and methods) and adding the redundant code. To improve this situation in the last section, a compression operation is done using the GZIP method. This stage is encrypted according to a series of functions. There are different methods for text compression but, the rate of compression operations and also obtaining the minimum compression is desired.

Implementation: The proposed method needed practical implementation for examination and evaluation which was developed using the Net toolset. This software benefits from a library file responsible for compression and decompression of the information. In this library file in addition to compression operations, encryption and decryption operations are done by a series of functions. The language of development tool is C# and performs the scan process on PHP codes in order to keep the final obtained result in an external file. The program user interface is observed in Fig. 2. The local section makes selecting an address from personal computer possible and remote mode provides the possibility of access and operating on the website or a remote place. The non-PHP files are not selectable and are only in display mode. The options selected in preference section change the algorithm's trend.

To implement the obfuscated code, it is required to record compressor and decompressor files in the system, so the PHP code can have access to it through API.

Test automation necessity: From 1996-2014, >25% of the all damages found in computer software were dependent on PHP (Coelho, 2015). In order to include the risks of the web's vulnerable applications, the penetration testers are employed to check the source code. Considering the fact that large applications can have thousands code lines and time is also limited to costs, the manual review of the source code may be incomplete. The tools can help the penetration testers to reduce the time and costs by automating the processes that depend heavily on time, when reviewing the resource code processes. In recent years, some tools have been introduced which can reduce the time required for the penetration tester by automating the identification process of potential security holes in PHP source using the analysis of static source code. So, the findings can be evaluated easily by the content penetration testers over again without re-examining the whole source code. But, considering the limitations of static source code, vulnerability analysis must be approved by a code reviewer. In the past, there were lots of application scanners under open source web, published with the aim of finding the vulnerabilities in the black box scenario by the fuzzy action. Review of source code in the white box scenario may lead to better results, but only a few open source PHP code analyzers are

Fig. 3: The results from RIPS software: a) Before securing the proposed software; b) After securing the proposed software

available. The results have shown that RIPS is able todetect the known and unknown security holes in PHP-based under web applications within the shortest time. Figure 3 shows an example of RIPS scan results before and after code securing. The proposed scheme has managed to enhance the security greatly.

For more studies on complex vulnerabilities, a list of the functions defined by the user is displayed in software user interface where the user can jump directly on the function code by clicking the title. Also, all of the functions defined by the user called as a result of scanning can be analyzed.

RESULTS AND DISCUSSION

Evaluation: The software compared to the proposed scheme have been mentioned in Table 1 with their parameters. The mentioned software are the most known software available in the market and some are presented commercially. In parameters section, the most important security attacks and factors were considered. Table 1 shows a list of the software using which the code has been obfuscated and their code output has been scanned in RIPS Software.

The source code is turned into an encrypted and obfuscated target code after entering the mentioned software and the target code is checked in RIPS software in terms of presence of security holes. Table 2 shows the results of this study. Also in order to study the performance of the proposed scheme in terms of volume of the generated files at the script execution time,we assess the efficiency improvement ratio of the target file and the security measures obtained from RIPS.

Size of the output files: Considering the encryption and steganography methods, size of the output files of the software mentioned in Table 1 will be different. Figure 4 implies that the size of the output files of out scheme takes the least host space. PHP Security is the same software developed based on the proposed method and primary code is the same basic code regardless of obfuscation and security enhancement. Although, size of the scripts is of no much importance compared to security enhancement, but producing a very high volume is also considered as a challenge. For example, FOPO changed the file volume to over 4 times more than the primary volume after encryption and obfuscation which is not optimal at all in terms of host consumption.

Runtime: Runtime of the program is always an important challenge in assessments. Using the updated approaches and algorithms can reduce the operation runtime. A code has been created to calculate the runtime of the screen. Figure 5 shows the runtime of the target codes created from security software.

Efficiency: In order to calculate the efficiency of the labour, it is just enough to divide the valueachieved from runtime by size of the file in order to obtain a value for each model or software.

Table 1: Comparison parameters of the proposed scheme with the similar software

Assessed parameters	Security evaluation	Security software	Script language
File inclusion, SQL injection, cross-site scripting, HTTP response splitting, possible flow control, file disclosure, code execution, obfuscation level, compatibility, sensitive sinks, improved security	RIPS	Zend Guard¡ FOPO¡ PHP Security¡ ionCube PHP Encoder¡ Gaijin¡ Code Eclipse æ Primary code	PHP

Table 2: Results from the Security Holes after obfuscation in different software

Software	Texe(mics)	Size (kb)	File inclusion	SQL injection	XSS	HTTP response splitting	Possible flow control	File disclosure	Code execution
Primary code	0.015	25.7	1	7	13	1	80	0	0
ionCube PHP encoder	0.090	53.7	0	0	3	0	2	0	0
PHP security	0.019	12.8	0	0	1	0	0	1	1
FOPO	0.014	111.0	0	0	0	0	0	0	2
Gaijin	0.017	32.9	0	7	13	1	0	0	6
Code eclipse	0.014	25.3	1	0	13	0	0	0	62
Zend guard	0.042	29.0	0	0	2	0	0	0	0

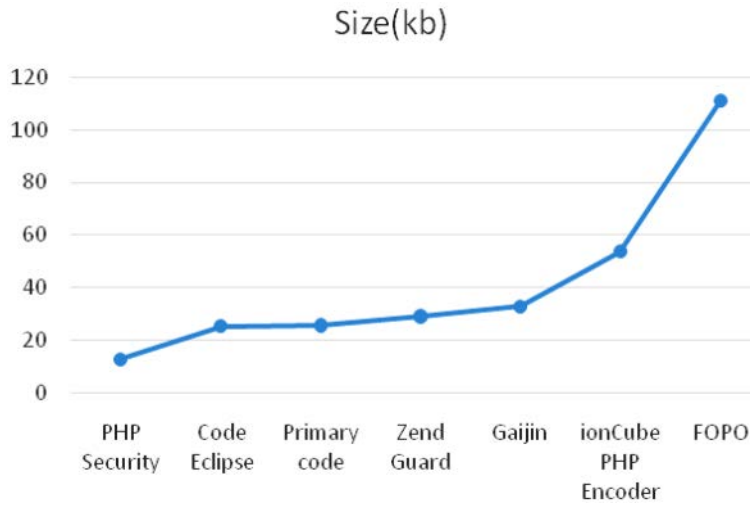

Fig. 4: Comparing the volume of output files

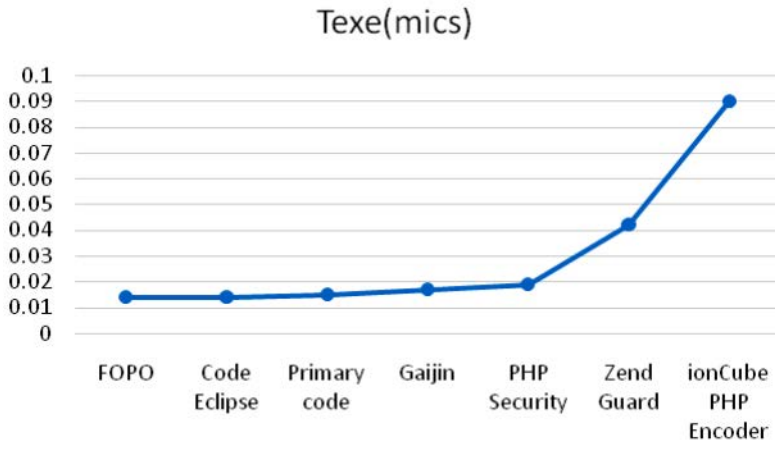

Fig. 5: Comparison of outputs in terms of runtime

As observed in Fig. 6, our idea has the most efficiency compared to its rivals. This efficiency can satisfy the costumers with the script running. The following relationship has been used to calculate efficiency. In this relationship, C_P is the efficiency rate, S_{SIZE} is the size of source file before obfuscation, S_{TEXE} is

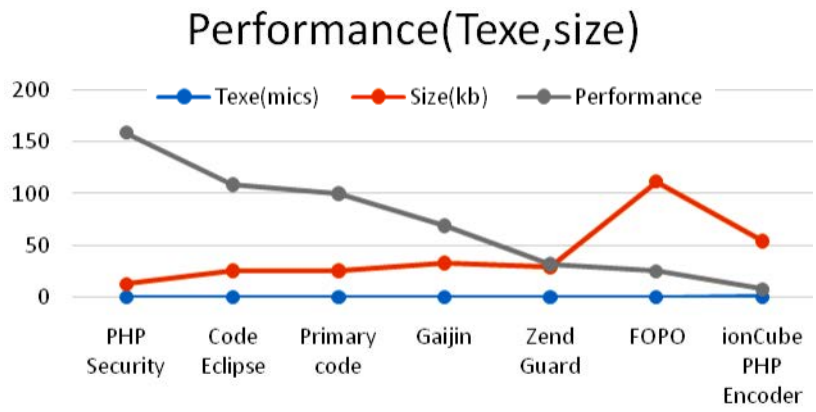

Fig. 6: Comparison of software efficiency in terms of runtime and file size

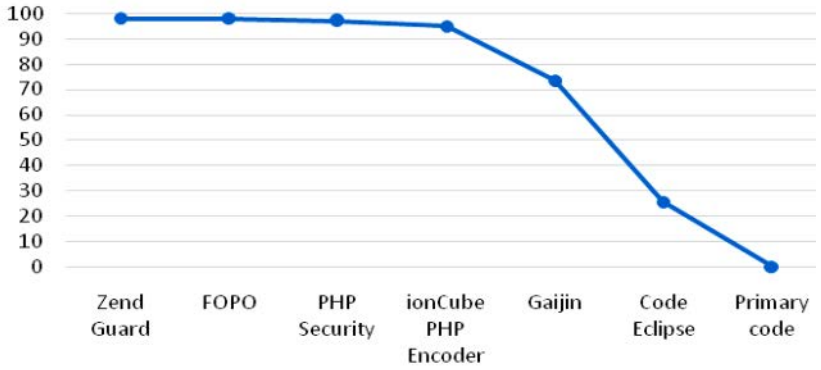

Fig. 7: Security enhancement comparison

Table 3: Security enhancement rate in security software

Software	Security enhancement (%)
Zend guard	98.03
FOPO	98.03
PHP security	97.05
ionCube PHP encoder	95.09
Gaijin	73.52
Code eclipse	25.49
Primary code	0.00

the runtime of the target file code and D_{TEXE} and D_{SIZE} respectivelyare the size and runtime of the target file:

$$C_P = ROUND ((S_{SIZE} \times S_{TEXE})/(D_{SIZE} \times D_{TEXE}) \times 100)$$

If the answer to the above equation, C_P, equals 100, it means that the obfuscated file is equal to target file in terms of runtime and file size. If it becomes more than 100, it means that the output file is even more optimal than the source file. As observed in Fig. 6, our scheme has obtained this efficiency.

Security enhancement: The most important part of comparison is the improved security discussion. According to the tests taken from different files each coded with different styles using the RIPS software, it was proved that PHP Security (the proposed idea) has a good position among the known software with a 97% enhancement of security. Table 3 shows the security improvement percentage by the proposed software in a descending order. To calculate the enhancement rate, the following relationship is established:

$$I_{SEC} = 100-((D_{SINKS} \times 100)/S_{SINKS})$$

Where:

I_{SEC} = The enhancement rate in percentage

D_{SINKS} = The number of holes found in the code obfuscated and tested in RIPS

S_{SINKS} = The hole numbers of the source code obtained in RIPS software

Figure 7 shows that the four zend, PHP Security, FOPO and IonCube software benefit from a close and good security level.

CONCLUSION

Obfuscation and encryption are two important processes for software security. This study aims at improving the safety and no access or costly access of the other people to the primary codes. Specifying the possible threads and attacks in the program, using the experts to identify the weaknesses in common codes, immunization plan of entry and exit points, adding the security extensions, encryption and compression are the most important processes in the implemented idea and software. At the end, the security test determines the accuracy of the work. In this study, we tested the server-side scripts to detect the holes by presenting a security scheme. In a few case studies, the obtained results in term of security and executive efficiency of this software show that software security has increased up to 97% compared to the primary code, keeping in the mind that the file size is decreased and the script runtime is not changed.

REFERENCES

Bayer, U., I. Habibi, D. Balzarotti, E. Kirda and C. Kruegel, 2009. A view on current malware behaviors. Proceedings of the 2nd USENIX Conference on Large-Scale Exploits and Emergent Threats: Botnets, Spyware, Worms and More, January 23, 2009, USENIX Association -.

Bisht, P. and V.N. Venkatakrishnan, 2008. XSS-GUARD: Precise dynamic prevention of cross-site scripting attacks. Proceedings of the International Conference on Detection of Intrusions and Malware and Vulnerability Assessment, July 10-11, 2008, France, pp: 23-43.

Brunton, F. and H. Nissenbaum, 2015. An Obfuscation Vocabulary. 1st Edn., MIT Press, Massachusetts.

Cholakov, N., 2008. On some drawbacks of the PHP platform. Proceedings of the 9th International Conference on Computer Systems and Technologies, June 2008, New York, USA -.

Coelho, F., 2015. PHP-related vulnerabilities on the national vulnerability database. http://www.coelho.net/php_cve.html.

Deepa, G. and P.S. Thilagam, 2016. Securing web applications from injection and logic vulnerabilities: Approaches and challenges. Inform. Software Technol., 74: 160-180.

Egele, M., T. Scholte, E. Kirda and C. Kruegel, 2012. A survey on automated dynamic malware-analysis techniques and tools. ACM Comput. Surveys, Vol. 44. 10.1145/2089125.2089126

Garcia-Alfaro, J. and G. Navarro-Arribas, 2007. Prevention of cross-site scripting attacks on current web applications. Proceedings of the OTM Confederated International Conferences on the Move to Meaningful Internet Systems, November 25-30, 2007, Portugal, pp: 1770-1784.

Gupta, S. and B.B. Gupta, 2016. Automated discovery of javascript code injection attacks in PHP web applications. Procedia Comput. Sci., 78: 82-87.

Kolbitsch, C., P.M. Comparetti, C. Kruegel, E. Kirda, X. Zhou and X.F. Wang, 2009. Effective and efficient malware detection at the end host. Proceedings of the 18th Conference on USENIX Security, December 14-18, 2009, India -.

Landesman, M., 2007. Malware revolution: A change in target. Microsoft Security Research and Response. https://technet.microsoft.com/en-us/library/cc512596.aspx.

Lanzi, A., D. Balzarotti, C. Kruegel, M. Christodorescu and E. Kirda, 2010. Accessminer: Using system-centric models for malware protection. Proceedings of the 17th ACM Conference on Computer and Communications Security, October 4-8, 2010, Chicago, IL., pp: 399-412.

Lu, H., X. Wang, B. Zhao, F. Wang and J. Su, 2013. ENDMal: An anti-obfuscation and collaborative malware detection system using syscall sequences. Math. Comput. Modell., 58: 1140-1154.

Mehrian, S.H.Z., S.A.R. Amrei and M. Maniat, 2016. Structural health monitoring using optimizing algorithms based on flexibility matrix approach and combination of natural frequencies and mode shapes. Int. J. Struct. Eng., Vol. 7, No. 4.

Mehrian, S.M.N. and S.Z. Mehrian, 2015. Modification of space truss vibration using piezoelectric actuator. Applied Mech. Mater., 811: 246-252.

Murtaza, S.S., W. Khreich, A. Hamou-Lhadj and A.B. Bener, 2016. Mining trends and patterns of software vulnerabilities. J. Syst. Software, 117: 218-228.

Schrittwieser, S., S. Katzenbeisser, P. Kieseberg, M. Huber, M. Leithner, M. Mulazzani and E. Weippl, 2014. Covert computation-hiding code in code through compile-time obfuscation. Comput. Secur., 42: 13-26.

Sharif, M., A. Lanzi, J. Giffin and W. Lee, 2008. Impeding malware analysis using conditional code obfuscation. Network and Distributed System Security (NDSS), http://llvm.org/pubs/2008-02-ImpedingMalwareAnalysis.pdf.

Tatroe, K., P. MacIntyre and R. Lerdorf, 2013. Programming PHP. O'Reilly Media, Sebastopol, California, ISBN: 9781449365844, Pages: 540.