EasyChair Preprint

№ 3713

A Vulnerability Detection Framework for CMS Using Port Scanning Technique

Md. Asaduzzaman, Proteeti Prova Rawshan, Nurun Nahar Liya, Muhmmad Nazrul Islam and Nishith Kumar Dutta

July 1, 2020

A Vulnerability Detection Framework for CMS Using Port Scanning Technique

Md. Asaduzzaman (✉), Proteeti Prova Rawshan, Nurun Nahar Liya, Muhmmad Nazrul Islam, and Nishith Kumar Dutta

Department of Computer Science and Engineering,
Military Institute of Science and Technology, Dhaka-1216, Bangladesh
asadbd45@gmail.com

Abstract. In the era of technology, attack on computer infrastructure is considered as the most severe threat. Web server is one of the most important components of this infrastructure. Preventive measures must be taken to deal with these attacks on the web servers. For this reason, vulnerability detection needs to be carried out in an effective way and should be mitigated as soon as possible. In this paper, an effective framework for vulnerability detection of web application is proposed. This framework targets the web applications developed with content management systems (CMSs). It obtains prior knowledge of the vulnerable extensions of a specific CMS from its contributors. The framework is run against a target web server using a well-known port scanning tool, Nmap. It checks if there is any existing matches for the vulnerable extension installed in that web application. Finally, the framework gives an output comprised of the installed extensions along with the installed vulnerable extensions in that web application. Although the output result is shown in the Nmap console, the framework is a segregated entity that works in collaboration with Nmap. Thus this framework can be well-utilized by the security specialists to assess the security of a web application in an easier and effective way and also to evaluate vulnerability of web servers; hence shielding the web applications from various kinds of security threats.

Keywords: Security scanner · Port scanning · Content management system · CMScan · Nmap Scripting Engine

1 Introduction

Nowadays, there is an increasing dependency on web applications. From an individual to an organization, almost every transaction is available, stored or traded in the web. Because of ease of access and its increasing nature of productivity and operational efficiency, reliability on web services has increased, which in turn has raised the security issue of the web applications. Web vulnerability refers to the system flaw or weakness through which the security can be compromised and resources can be exploited. Attacker can access the flaw; thereafter breach the system integrity through exploitation. This can be easily detected by using network vulnerability scanners, which identify the security loopholes in a computer

network by inspecting the most potential targets. Network vulnerability scanners like: SARA [1], SAINT [2], VLAD [3] and Nessus[4] are very effective but most of them are paid and require technical knowledge to use. Whereas, Nmap[5] is a multipurpose utility tool and a port scanner, which is used by millions of beginner users for its easy usability. It discovers services and hosts running in a computer network, including host and port discovery. An NSE script [6] allows doing a wide variety of network assessment tasks.

A widely used application for managing web contents is the Content Management System (CMS). It supports a modular and adaptable framework with the installation of plugins, so that new features can be added and thus the main functionalities of the CMS can be achieved. Amongst all, the most widely used CMS platforms are: WordPress (58.8%), WeBex (12%), Joomla (6.5%) and Drupal (4.8%) [7]. Kaluža et al. [8] carried out a survey on a number of companies and found that 61.11% of the companies used CMS, where 48.48% of the respondents used free CMS, 6.06% answered commercial, 18.18% answered custom CMS and 27.27% of the respondents failed to provide an answer. The CMSs can be kept secured if all the extensions and the plugins can be updated regularly. But the most common problem is that amongst the huge number of plugins, maximum are getting outdated thus compatibility issues are created while using the latest versions.

The main vulnerability issue of CMS lies within its feature-easy identification. Outdated plugins are the entry points for most of the attackers. Cernica et al. [9] showed that from the top ten million websites, 16% of them used WordPress. The paper also conveys that from the total of 21 backup plugins, 12 were found to be vulnerable that can lead to 'Sensitive Data Exposure'. Martinez-Caro et al. [10] conducted an extensive study on CMS alongside some basic security analysis on Joomla and Drupal and found some security vulnerabilities in the extensions of Joomla and Drupal which can be dangerous. Studies show that in 2018, 90% of the hacked CMS based websites used WordPress, then Magento taking up to 4.6% of the data sample, 4.3% of the websites with joomla then consecutively Drupal and ModX [11]. With these kinds of publicly disclosed exposures, it is easier for the attackers to exploit. Network security professionals often have to depend on the other paid vulnerability assessment tools in order to assess the security of web applications (including CMS). Besides, almost all network-security professionals along with network administrators are experts on using open source port scanners. So, an advanced framework can be incorporated in the port scanner that will allow the users to assess vulnerabilities of CMSs.

Therefore, the objective of this paper is to integrate the most required functionalities of a vulnerability scanner for CMSs with a popular port scanner. In order to attain this objective, this research proposes to build an open source framework which incorporates an NSE script in a port scanner (Nmap). It can detect the installed extensions in a CMS; hence it can detect the vulnerable extensions along with the affected versions.

The remaining sections of this paper are organized as follows: a brief overview of the related work is presented in section 2, the conceptual framework is dis-

cussed in section 3. In section 4, the design and development of the framework is discussed. Further, the evaluation of the framework is presented in section 5, followed by a discussion and conclusion in section 6.

2 Literature Review

This research focused on the field of CMS based web applications, their vulnerabilities, security aspects and contextual threats and also the ways they can be exploited. To find out the related literature, a search was conducted in the major scholar databases including ACM Scholar, Google Scholar, IEEE Explorer and ScienceDirect using suitable search strings. The related literatures are presented briefly below.

Most of the CMSs are customizable, adaptable and built-in open source frameworks (WordPress, Joomla or Drupal) [12], hence they are vulnerable by their nature. Also, a shared environment provides the users with shared flaws which encourages the security researchers and the hacker community. Once these vulnerable loopholes are found, they are used for mass attacks. Yu et al. [13] made a model of mapping these vulnerabilities and attack patterns by analyzing the attack targets. He also developed a methodology to test and detect them in web services. Scott et al.[14] introduced a Secured Web Applications Project(SWAP) against various application level attacks. It protects against a large class of attacks than existing web methodologies. In addition, Kals et al. [15] proposed SecuBat, another vulnerability scanner to analyze web sites for exploitable SQL and XSS vulnerabilities.

As the most common format of exploit is SQL injections, Wassermann et al. [16] approached an automated and precise solution. It characterizes the values of string variable assuming with a context free grammar and tracks the user modifiable data by modeling string operations. It is implemented in PHP, discovers both known and unknown vulnerabilities as well as scales to large sized programs. Huang et al.[17] created a static analysis algorithm and a tool named WebSSARI, which statistically verifies CMSs' code where run time overhead is reduced to zero with sufficient annotations. After verifying, it automatically secures the potentially vulnerable sections of the code. Jovanovic et al.[18] introduced another static analysis tool(Pixy). For detecting XSS vulnerabilities in PHP, as well as detecting taint-style algorithms like SQL or command injections Pixy uses data-flow analysis and is written in Java. Fu et al.[19] proposed another static analysis tool which automatically generates test cases exploiting SQL injection vulnerabilities in ASP.NET web applications.

Few researches are conducted using Nmap NSE scripts. Rosa et al. [20] developed a number of open-source tools for analysis of PCOM security aspects that includes a Nmap NSE PCOM scan. In [21], Nmap NSE is used for testing authentication servers for malware infection. But no research is conducted for the CMS scan.

There is a number of existing Nmap NSE scripts that serve different purposes during vulnerability assessment [22]. Two of the scripts named *http-wordpress-*

enum.nse and *http-drupal-enum.nse* can be used to detect the vulnerabilities of websites that are developed with WordPress and Drupal [23]. These scripts only allow users to detect vulnerabilities of WordPress (*http-wordpress-enum.nse*) and Drupal (*http-drupal-enum.nse*) respectively based on a limited number of extensions listed in *wp-plugins.lst*, *wp-themes.lst*, *drupal-modules.lst* and *drupal-themes.lst* [24]. But these are two different scripts and unable to accommodate new CMS.

In sum, though there are numerous existing methods of detecting vulnerabilities of web based applications, almost all of them are paid. The most required functionalities of vulnerability scanner and port scanner are not integrated together yet. Although some functionalities are integrated, these only cover two specific CMSs. Thus this research work will focus to develop an open-source framework that will achieve these features using port scanning technique in the context of CMSs.

3 Conceptual Framework

The proposed conceptual framework for vulnerability detection is depicted in Figure 1. The whole design process consists of two stages: Information Gathering Stage and Operational Stage. In the Information Gathering Stage, informational details about a new CMS will be collected. Based on the information achieved, the main framework will be run to detect the vulnerabilities during the Operational Stage.

One of the major concerns is to accurately detect the vulnerable extensions or vulnerable core CMS and to minimize the security risks. Another concern is to adapt a newly developed CMS in the framework. It is needed to enrich the list of information of the CMSs in order to maximize the accuracy. So, the repository is made public so that contributors can enrich the information of existing CMS and append information about a new CMS in the Information Gathering Stage. Each of the contribution will be highly appreciated in the contribution section. The information contains details (i.e CMSs' name and common directory structure for the extensions) list of extensions available to install and list of vulnerable extensions which are publicly available in JSON format. Conventions to contribute in the repository are documented in the development process.

In the Operational Stage, the framework is run against a web server (the target host). The framework constructs a URL using the directory structure which resides in the aforementioned JSON file. It checks for the existence of the directory (HTTP response: 200) and takes decision accordingly. If the directory exists, it goes for further operation. Following the similar process, extensions are extracted from the web server those are already installed in the CMS. Vulnerability checking of this CMS is performed by analyzing the installed extensions with respect to the vulnerable extensions' list.

The scanner returns a list of vulnerable extensions. It can also return the affected versions, provided that the installed extension and vulnerable extension contain version information.

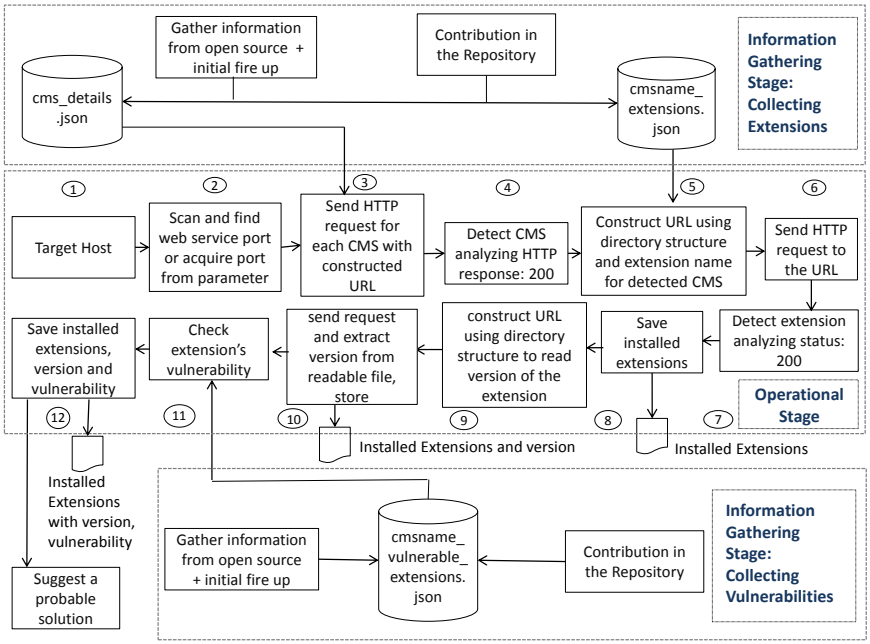

Fig. 1. Working process of the proposed framework

4 Development of the Framework

The framework can accommodate any newly released CMS. This framework works in two phases- *firstly*, it gathers the list of CMSs' details along with the list of all the extensions and vulnerable extensions that is referred to as the *Information Gathering Stage*. *Secondly*, a website is scanned with vulnerability detection framework based on the gathered information in the first stage, using a port scanning technique (*Nmap*) that is referred to as the *Operational Stage*.

Information Gathering Stage This part of the framework is open to all the contributors. Any contribution to this open source tool will be highly appreciated in the contributors section of GitHub. Initially, the GitHub repository contains information of WordPress and Joomla which can be updated through the proper pull requests and verification. Anytime, new CMSs can be added to the repository by appending the lists of information by following the instructed conventions. There is a *cms_details.json* file which contains the details about the CMS in an array. In order to append a new CMS in the framework, the *cms_details.json* must be updated with the new CMS name, CMS version file directory and array of CMS extension file directory. A new CMS entry can be appended to the array according to the following syntax:

```
[{
  "cms_name": "example",
  "cms_version_file_directory": "/example.txt",
  "cms_extension_directory": [
      "/directory_1",
      "/directory_2"
    ]
}]
```

The cms_version_directory is the directory path of a file that contains version of the installed CMS and the cms_plugin _directory contains the array of the directories those contain the installed extensions.

There are two other files to be created for each CMS. One is the *cmsname_extension .json* that contains all the extensions available for installation in the *cmsname* CMS using the following syntax:

```
[{
  "extension": ["example_1","example_2"],
  "directory": ["dir_1/{ext}/{ext}.txt",
  "dir_2/{ext}/{ext}.txt"]
}]
```

Here, *directory* denotes the file paths those contain version of the extensions. In the Operational Stage *ext* will be replaced with the extension name. Another file to be created for each CMS with the name *cmsname_vulnerable_extensions.json* that contains all the vulnerable extensions along with the list of affected versions for *cmsname* CMS using the following syntax:

```
[{
    "vulnerable_plugin":"vulnerable_plugin_name",
    "affected_version":["4.5","2.1"],
    "description_of_vulnerability":"SQLInjection",
    "code_type":["cve","exploitdb"],
    "code":[{"cve":"xxxxx","exploitdb":"xxxxx"}],
    "source_url":["https://example.com/xxxx/",
    "https://example.com/xxxx"]
    }]
```

The above syntax may be changed and will be updated accordingly in the GitHub documentation in case of any change in the framework.

Operational Stage In this stage, the framework works like an operational tool using the gathered information from the previous stage. In this paper, the Operational Stage for the proposed framework is developed with an Nmap script written in *Lua programming language*. Name of the script is *cmscan.nse*. When the *nmap –script cmscan target* command is run in the terminal, the script is called and it starts working. At first, it detects the CMS looking into the

cms_details.json and thereby recognizing the directory structure. It reads the version file, provided that the file is available in order to check the vulnerability of the core CMS. Then it takes the directory path of installed extensions from the same file. It looks for the *cmsname_extension.json* file to check an extension's existence in the CMS. A URL is constructed from the directory path of extensions and it is appended after the host. Then it checks for the URL's existence using the *http request*. If the extension exists in the CMS, the version is extracted by reading the file, a URL is constructed using the directory given, that looks up on the *cmsname_vulnerable_extensions.json* file to check whether any version of the extension is vulnerable or not. If the version is not found, it is suggested that the user should look for the extension manually.

In this way, the vulnerable plugins are detected along with its CVE, description of the vulnerability and source URL. The process is made faster and efficient using the concept of multi-threading and parallelism. Initially the framework has a base of huge list of information that contains two CMSs: WordPress and Joomla for the initial fire up. However, more CMSs can be accommodated in the framework.

5 Performance Evaluation of the Framework

To evaluate the performance of the framework, a simple experiment is conducted in two phases. Firstly, two web servers are set up with two different CMSs (i.e WordPress and Joomla) to evaluate the performance. Secondly, the framework is run against a number of web servers within a private network, where the servers are mostly operated with CMSs. In the first phase, WordPress and Joomla are installed in two different servers. Some extensions are installed in both of the servers. Some vulnerable extensions are installed in the servers intentionally. The two applications are hosted in the servers with IPs *192.168.0.10* and *192.168.0.12* for Joomla and WordPress respectively. The Nmap scan is performed against these two IP addresses by running the following commands in terminal-

```
nmap --script http-cmscan -p80 192.168.0.10
nmap --script http-cmscan -p80 192.168.0.12
```

Figure 2 shows the scan result for WordPress based web application. The scan result finds 6 plugins, 2 of the plugins are vulnerable. The names of the vulnerable plugins are *loco-translate 2.2.1* and *wp-cerber 8.0*. The scan returns result in 1.25 seconds. While figure 3 shows the installed plugins and plugin details from the WordPress admin panel. Another snapshot of scan result for Joomla based web application is depicted in figure 4 and the extensions page of Joomla admin panel is depicted in figure 5. In this scenario four extensions are found in the server, one of the extensions is found to be vulnerable.

In the next phase, the script is run against a block of IP address (172.16.0.0/24) where a number of CMSs are hosted. Nine hosts are found those run CMSs in

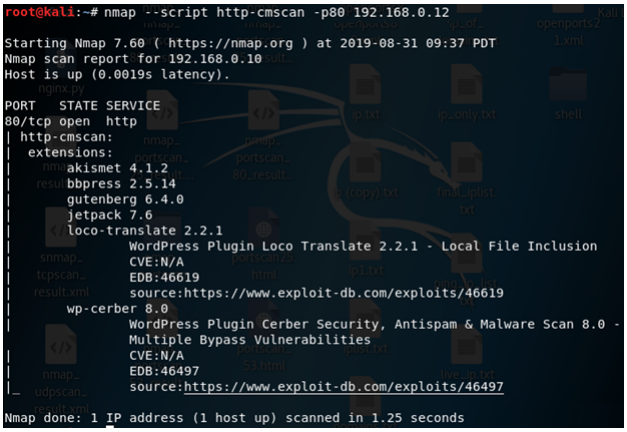

Fig. 2. Passive scan result of the WordPress host

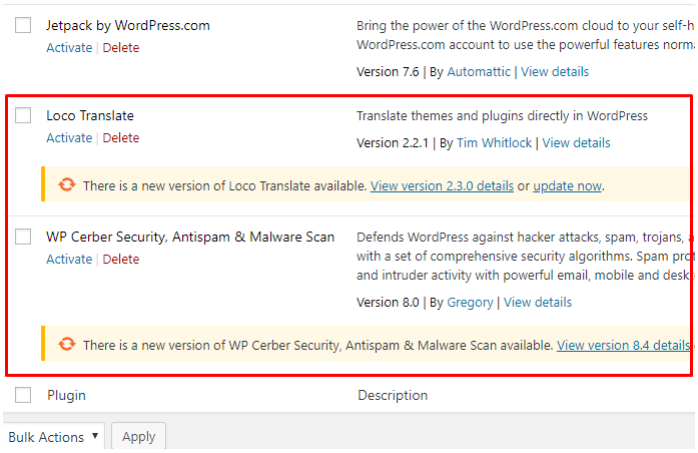

Fig. 3. WordPress extension page from admin panel

web server, most of the servers are run with WordPress. The result summary is given in the Table 1.

As per Table 1, the framework detects nine web servers. Two of the servers are using Joomla CMS and another seven servers are using WordPress CMS. Total number of plugins is shown in the #plugins column and total number of vulnerable plugins is shown in the #vulnerability column. Required time for the scan process is also shown for the different target IPs. The framework gives output based on the extension's information list. If the list is rich and accurate, the output will be accurate, otherwise it can be misled. The output delay depends on the number of extensions installed in the target web application and on the link speed. The result shows that using the stated port scanning technique, the framework can work efficiently and accurately based on the information list.

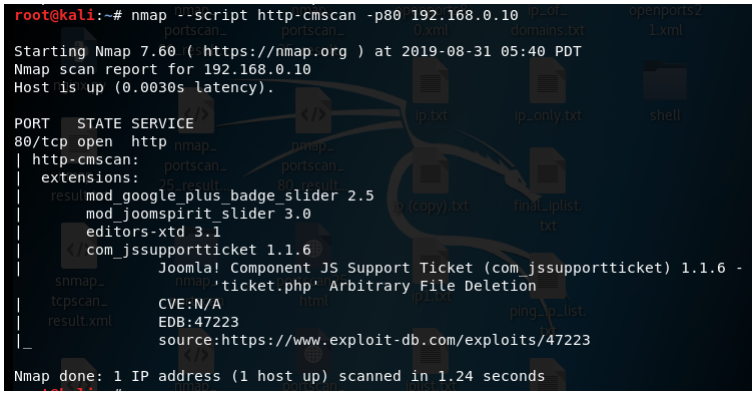

Fig. 4. Passive scan result of the Joomla host

Status	Name	Location	Type	Version	Date	Author	Folder
✓	Ultimate Google Plus Badges Slider	Site	Module	1.0	September, 2013	Bill Bachez	N/A
✓	JoomSpirit Slider	Site	Module	2.9	January 2016	JoomSpirit	N/A
✗	Editor Button - GroupDocs Signature	Site	Plugin	1.4.2	March 2015	GroupDocs Team	editors-xtd
✓	JS Support Ticket	Administrator	Component	1.1.7	Aug 11th, 2019	Joom Sky	N/A
✓	Feed Display	Administrator	Module	3.0.0	July 2005	Joomla! Project	N/A
✗	JS Support Ticket icon	Site	Plugin	1.0.1	October 29th, 2015	Joom Sky	system

Fig. 5. Joomla extension page from admin panel

Thus the proposed framework will help the security specialists to figure out the serious vulnerabilities which are potential to cause huge damages.

Table 1. Sample time and vulnerability of target websites.

Target	CMS	Time(Sec)	#plugins	#vulnerability
172.16.0.12	Wordpress	24.08	5	1
172.16.0.13	Wordpress	27.11	10	2
172.16.0.32	Joomla	257.14	98	14
172.16.0.33	Joomla	165.75	61	11
172.16.0.34	Wordpress	100.45	47	5
172.16.0.42	Wordpress	25.79	6	0
172.16.0.78	Wordpress	59.98	21	1
172.16.0.74	Wordpress	41.841	13	0
172.16.0.61	Wordpress	28.44	8	1

6 Discussion and Conclusions

In this paper, a framework is proposed that integrates the most important components of a vulnerability scanner with a port scanner in the context of CMS. Knowledge base of this framework is CMSs' information which is mostly dependent on the contributors. But the information will be updated from servers as well, which minimizes the framework's dependency on the contributors. As a result the framework will help in vulnerability assessment by detecting the vulnerabilities of a CMS efficiently.

The main implication of the framework is that it requires less effort to operate. Also, it is not needed to go through the hassle of paid and full-fledged vulnerability scanners. The network administrator can also use this to know about the possible vulnerabilities.

There are a number of existing tools to serve the purpose of vulnerability assessment. Most of the tools are heavy and paid. There is a shortage of open source tools that can help to assess the vulnerabilities. Most of the open source tools are not dedicated for the CMSs and so fails to detect the vulnerabilities of most of the CMSs. These tools are good for only specific CMSs. Although Nmap is a popular multipurpose tool for vulnerability assessment, there is no script or framework of Nmap to assess the vulnerabilities of CMSs [22]. In this paper, an open source framework is proposed that can be used for the vulnerability assessment of all the CMSs using Nmap. The framework serves the purpose of vulnerability assessment for a broader range of websites developed with CMSs.

The framework is currently being operated using port scanning technique and is dependent on Nmap. Also the knowledge base of this framework is mostly dependent on the contributors. The run time of the framework varies with the configuration of machine and network connectivity with the target host. The machine needs internet connection to perform the scan.
In future, the main initiative is to make the framework independent and as well as to incorporate in the other popular security tools. Also the aim is to minimize the dependency on the contributors by deploying servers for the purpose of gathering and updating the information about CMS. The scan can also be performed without internet connection; in that case the information lists are to be downloaded to the local machine in the same directory of the script. This process does not ensure the updated repository to be resided in the user's machine. Although a number of network scanning tools exist in the open source, but few of those are developed for CMSs scan. Also the tools are developed for a specific CMS. Some tools have support to scan all kind of CMSs, but the users are to pay a heavy cost for the tools. Performance evaluation can be carried out by comparing the output for a specific CMS with the existing open source tools and also with the paid tools. In future, a detailed performance evaluation and comparison will be conducted with the existing frameworks.

In this new course of technological evolution where everyone uses devices which is more or less connected to common or private networks. Access, misuse and hacking of files and directories are happening more than ever. The framework can help to find these vulnerabilities and detect the ways through which network

interrogation is possible to inform the users or the administrator, thus protecting from further attacks by making a more integrated and rigid network.

References

1. Security auditor's research assistant, http://www-arc.com/sara/. Last accessed 29 Nov 2019
2. Saint cybersecurity solution, http://www.saintcorporation.com/. Last accessed 29 Nov 2017
3. Vlad the scanner, http://www.decuslib.com/decus/vmslt00b/net/vlad_readme.html. Last accessed 29 Nov 2017
4. Nessus vulnerability scanner, https://www.tenable.com/products/nessus-vulnerability-scanner. Last accessed 29 Nov 2017
5. Lyon, G.F.: Nmap network scanning: The official Nmap project guide to network discovery and security scanning. Insecure (2009)
6. Nse-nmap scripting engine, https://nmap.org/book/nse.html. Last accessed 29 Nov 2017
7. Market share:top website platforms and example sites, https://websitesetup.org/popular-cms/. Last accessed 29 Nov 2017
8. Kaluža, M., Vukelić, B., Rojko, T.: Content management system security. Zbornik Veleučilišta u Rijeci **4**(1), 29–44 (2016)
9. Cernica, I.C., Popescu, N., Tiganoaia, B.: Security evaluation of wordpress backup plugins. pp. 312–316 (05 2019). https://doi.org/10.1109/CSCS.2019.00056
10. Martinez-Caro, J.M., Aledo-Hernández, A.J., Guillen-Perez, A., Sanchez-Iborra, R., Cano, M.D.: A comparative study of web content management systems. Information **9**, 27 (01 2018). https://doi.org/10.3390/info9020027
11. Website hacked trend report 2018, https://sucuri.net/reports/19-sucuri-2018-hacked-report.pdf. Last accessed: 24 Jan 2020
12. Meike, M., Sametinger, J., Wiesauer, A.: Security in open source web content management systems. IEEE Security & Privacy **7**(4) (2009)
13. Yu, W.D., Aravind, D., Supthaweesuk, P.: Software vulnerability analysis for web services software systems. In: Computers and Communications, 2006. ISCC'06. Proceedings. 11th IEEE Symposium on. pp. 740–748. IEEE (2006)
14. Scott, D., Sharp, R.: Developing secure web applications. IEEE Internet Computing **6**(6), 38–45 (2002)
15. Kals, S., Kirda, E., Kruegel, C., Jovanovic, N.: Secubat: a web vulnerability scanner. In: Proceedings of the 15th international conference on World Wide Web. pp. 247–256. ACM (2006)
16. Wassermann, G., Su, Z.: Sound and precise analysis of web applications for injection vulnerabilities. In: ACM Sigplan Notices. vol. 42, pp. 32–41. ACM (2007)
17. Huang, Y.W., Yu, F., Hang, C., Tsai, C.H., Lee, D.T., Kuo, S.Y.: Securing web application code by static analysis and runtime protection. In: Proceedings of the 13th international conference on World Wide Web. pp. 40–52. ACM (2004)
18. Jovanovic, N., Kruegel, C., Kirda, E.: Pixy: A static analysis tool for detecting web application vulnerabilities. In: Security and Privacy, 2006 IEEE Symposium on. pp. 6–pp. IEEE (2006)
19. Fu, X., Lu, X., Peltsverger, B., Chen, S., Qian, K., Tao, L.: A static analysis framework for detecting sql injection vulnerabilities. In: Computer Software and Applications Conference, 2007. COMPSAC 2007. 31st Annual International. vol. 1, pp. 87–96. IEEE (2007)

20. Rosa, L., Borges de Freitas, M., mazo, s., Monteiro, E., Cruz, T., Simoes, P.: A comprehensive security analysis of a scada protocol: from osint to mitigation. IEEE Access **7** (03 2019). https://doi.org/10.1109/ACCESS.2019.2906926
21. Basam, D., Ransbottom, J., Marchany, R., Tront, J.: Strengthening mt6d defenses with lxc-based honeypot capabilities. Journal of Electrical and Computer Engineering **2016**, 1–13 (01 2016). https://doi.org/10.1155/2016/5212314
22. Rahalkar, S.: Introduction to nmap. In: Quick Start Guide to Penetration Testing, pp. 20–39. Springer (2019)
23. Rahalkar, S.: Introduction to nmap. In: Quick Start Guide to Penetration Testing, p. 23. Springer (2019)
24. List of data in nse libraries, https://svn.nmap.org/nmap/nselib/data/. Last accessed 04 Sept 2019