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. Vulnerability testing of networks and vulnerability prevention are crucial and required tasks in today's world due to the increasing rate of the number of attackers and intruders. Various scanners can help the administrators to perform these tasks. This paper aims to propose an architecture of an open source framework to efficiently detect the vulnerabilities of websites developed with Content Management Systems (CMSs) using a port scanning technique. The proposed framework can detect vulnerabilities based on the list of vulnerable extensions and vulnerable core CMSs which are publicly available. The framework can accommodate any kind of CMS. Developers only need to add the information of a new CMS in the lists. As accuracy depends on the information of extensions, the extension list was made public to the contributors so that they can append or update the information. A large information list of two CMSs has already been prepared for the initial fire up.

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. Reliability of web services has occurred because of ease of access and its increasing nature of productivity and operational efficiency, which in turn raised the security issue of the web applications, that basically lies on website security and their vulnerabilities. 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 reduces the system integrity through exploitation. This can be prevented easily by using network vulnerability scanners, that identifies the security loopholes in a computer network by inspecting the most potential targets. Network vulnerability scanners like: SARA [10], SAINT [9], VLAD [11] and Nessus [6] are very effective but most of them are paid and require technical knowledge to use. Whereas, Nmap [7] is a multipurpose utility tool and a port scanner, which is used by millions of novice users and can be used easily. It discovers services and hosts running in a computer network, including host and port discovery. An NSE script [8] allows to do a wide variety of network assessment tasks and customizes them.

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 by additional extensibility. Among many, mostly used CMS platforms are: WordPress (58.8%), WeBex (12%), Joomla (6.5%) and Drupal (4.8%) [5]. They 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 amount of plugins, maximum are getting out of date thus compatibility issues are created with the latest versions.

The main vulnerability issue of CMS lies within its feature-easy identification. Then followed by outdated plugins which is the point of entry for most of the attackers, using poor and reused codes, lack of observation of the system administrator while giving regular updates and lack of CMS installations. The percentage of running out-of-date plugins in the compromised sites are the biggest problem. Studies show that the 78% of the hacked websites used WordPress, then Joomla taking up to 14% of the data sample, 6% with websites with raw codes then consecutively Magneto and Drupal [12]. With these kind of publicly disclosed vulnerabilities, it is easier for the attackers to exploit. Network security professionals often are to depend on the other paid vulnerability assessment tools in order to asses the vulnerabilities of web applications(including CMS). Besides almost all network security professionals along with network administrators are expert on using open source port scanner. So, an advanced framework can be incorporated in the port scanner that will allow users to assess vulnerabilities of Content Management Systems.

Therefore, the objective of this paper is to integrate the most required functionalities of a vulnerability scanner for the Content Management Systems based on the vulnerable extensions along with a popular port scanner. In order to attain the objective, this research proposes to build an open source framework which incorporates an NSE script in a port scanner(Nmap) that can detect the installed extensions in a Content Management System, hence 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 discussed 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 to the field of CMS based web applications, their vulnerabilities, security aspects and contextual threats and 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 literature are presented briefly below.

Most of the CMSs are customizable, adaptable and built in open source frameworks (WordPress, Joomla or Drupal) [17], 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. [20] 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. [18] 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. [16] 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. [19] approached an automated and precise solution. It characterizes the values of string variable assuming with a context free grammar hence tracking 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. [14] 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. [15] 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. [13] proposed another static analysis tool which automatically generates test cases exploiting SQL injection vulnerabilities in ASP.NET web applications.

Nmap has it's own scripts as well which can be used to detect the vulnerabilities of wordpress and drupal, name of the scripts are *http-wordpress-enum.nse* and *http-drupal-enum.nse* respectively. These scripts only allow users to detect vulnerabilities of wordpress [3] and drupal [2] respectively based on a limited number of extensions listed in *wp-plugins.lst*, *wp-themes.lst*, *drupal-modules.lst* and *drupal-themes.lst* [4]. But these are two different script 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 which will achieve these features using port scanning technique in the context of any Content Management Systems.

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 along with minimizing the security risks hence it is required to protect the system using port scanning technique. Another concern is to adapt a newly developed CMS in the framework. It is needed to enrich the information list 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 during 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, which can also be found in the GitHub repository documentation [1].

In the Operational Stage, the framework is run against a web server (the target host). The framework constructs a URL pointing to 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 which are already installed in the CMS. Vulnerability checking of this CMS is performed by analyzing the installed extensions with respect to the vulnerable extension's 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.

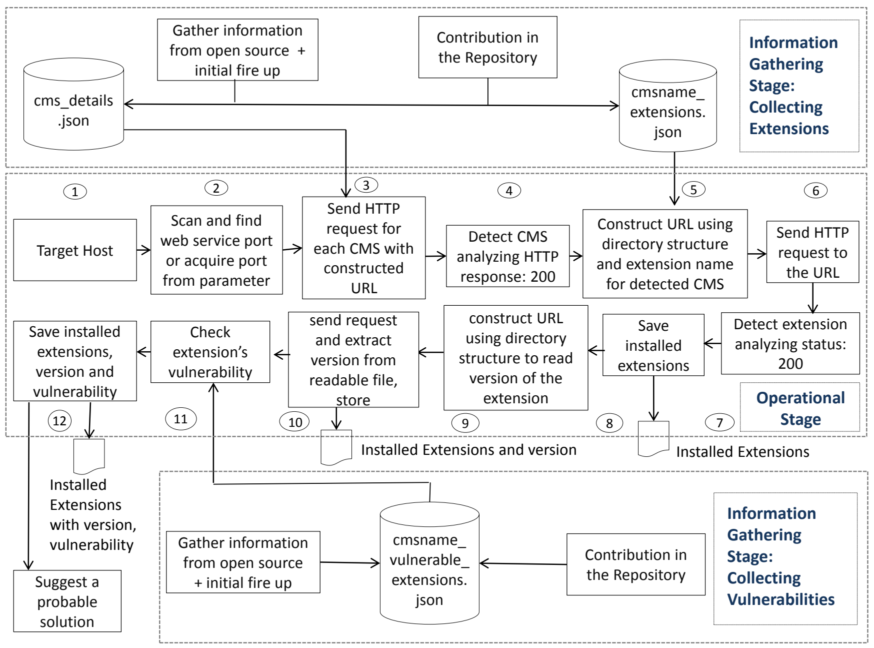

Gather information from open source + initial fire up
Contribution in the Repository
cms_details .json
cmsname_ extensions. json
Information Gathering Stage: Collecting Extensions
1
2
3
4
5
6
Target Host
Scan and find web service port or acquire port from parameter
Send HTTP request for each CMS with constructed URL
Detect CMS analyzing HTTP response: 200
Construct URL using directory structure and extension name for detected CMS
Send HTTP request to the URL
Save installed extensions, version and vulnerability
Check extension's vulnerability
send request and extract version from readable file, store
construct URL using directory structure to read version of the extension
Save installed extensions
Detect extension analyzing status: 200
Operational Stage
12
11
10
9
8
7
Installed Extensions and version
Installed Extensions
Installed Extensions with version, vulnerability
Suggest a probable solution
Gather information from open source + initial fire up
cmsname_ vulnerable_ extensions. json
Contribution in the Repository
Information Gathering Stage: Collecting Vulnerabilities

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 CMS' details along with the list of all the extensions which can be installed with the list of the vulnerable extensions as refers as the *Information Gathering Stage*. *Secondly*, a website is scanned with vulnerability detection based on the information gathered in the first stage, using a port scanning technique(*Nmap*) as referred 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"]
    }]
```

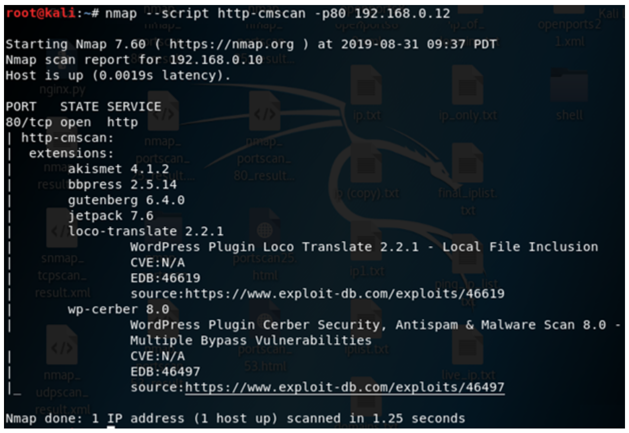
The above syntax may be changed and will be updated accordingly in the GitHub documentation in case any change in the framework.

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 as 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 fired 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 when it's appended after the host. Then it checks for the URL's existence using the *http request*. If the extension exist in the CMS, the version is extracted by reading the file , URL is constructed using the directory given. That checks 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 looks for the extension manually.

In this way, the vulnerable plugins are detected along with it's cve code, description of the vulnerability, 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 contain 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 setup 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. Whereas, 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-

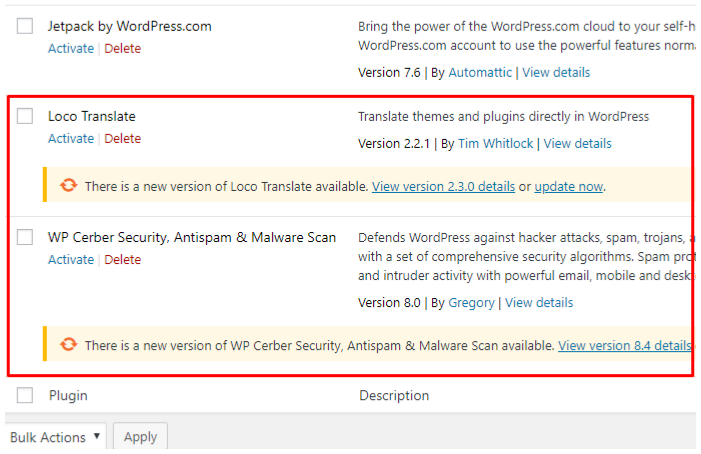


```
root@kali:~# nmap --script http-cmscan -p80 192.168.0.12

Starting Nmap 7.60 ( https://nmap.org ) at 2019-08-31 09:37 PDT
Nmap scan report for 192.168.0.10
Host is up (0.0019s latency).

PORT   STATE SERVICE
80/tcp open  http
| http-cmscan:
|  extensions:
|       akismet 4.1.2
|       bbpress 2.5.14
|       gutenberg 6.4.0
|       jetpack 7.6
|       loco-translate 2.2.1
|               WordPress Plugin Loco Translate 2.2.1 - Local File Inclusion
|               CVE:N/A
|               EDB:46619
|               source:https://www.exploit-db.com/exploits/46619
|       wp-cerber 8.0
|               WordPress Plugin Cerber Security, Antispam & Malware Scan 8.0 -
                Multiple Bypass Vulnerabilities
|               CVE:N/A
|               EDB:46497
|_              source:https://www.exploit-db.com/exploits/46497

Nmap done: 1 IP address (1 host up) scanned in 1.25 seconds
```

Fig. 2. Passive scan result of the wordpress host

```
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 name 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.

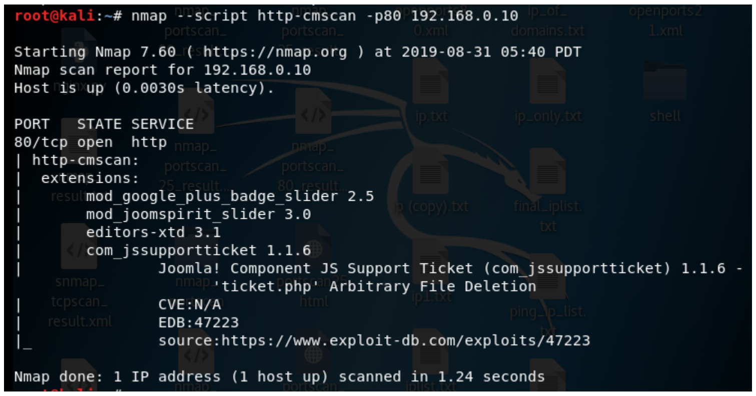


Jetpack by WordPress.com
Activate | Delete
Bring the power of the WordPress.com cloud to your self-h
WordPress.com account to use the powerful features norm
Version 7.6 | By Automattic | View details

Loco Translate
Activate | Delete
Translate themes and plugins directly in WordPress
Version 2.2.1 | By Tim Whitlock | View details
There is a new version of Loco Translate available. View version 2.3.0 details or update now.

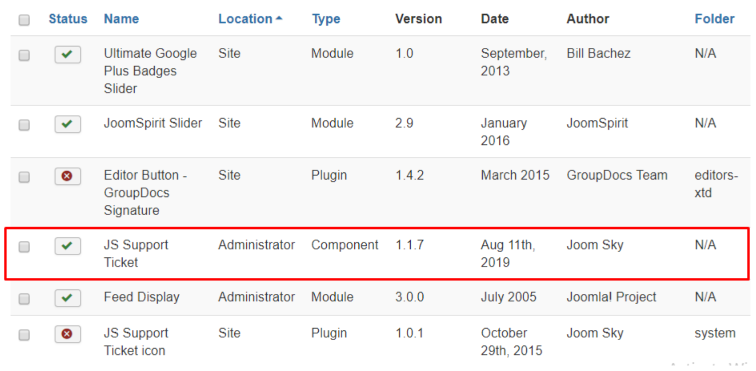
WP Cerber Security, Antispam & Malware Scan
Activate | Delete
Defends WordPress against hacker attacks, spam, trojans,
with a set of comprehensive security algorithms. Spam pr
and intruder activity with powerful email, mobile and desk
Version 8.0 | By Gregory | View details
There is a new version of WP Cerber Security, Antispam & Malware Scan available. View version 8.4 details

Fig. 3. Wordpress extension page from admin panel



```
root@kali:~# nmap --script http-cmscan -p80 192.168.0.10

Starting Nmap 7.60 ( https://nmap.org ) at 2019-08-31 05:40 PDT
Nmap scan report for 192.168.0.10
Host is up (0.0030s latency).

PORT   STATE SERVICE
80/tcp open  http
| http-cmscan:
|   extensions:
|        mod_google_plus_badge_slider 2.5
|        mod_joomspirit_slider 3.0
|        editors-xtd 3.1
|        com_jssupportticket 1.1.6
|                Joomla! Component JS Support Ticket (com_jssupportticket) 1.1.6 -
|                       'ticket.php' Arbitrary File Deletion
|                CVE:N/A
|                EDB:47223
|_               source:https://www.exploit-db.com/exploits/47223

Nmap done: 1 IP address (1 host up) scanned in 1.24 seconds
```

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

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 web

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

server, most of the servers are run with Wordpress. The result summary is given in the Table 1. The result shows that using the stated port scanning technique, the framework can work efficiently and accurately based on the information list. Thus the proposed framework will help the security specialists to figure out the serious vulnerabilities which are potential to cause huge damages.

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.

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.

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 is happening more than ever. The framework can help 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. Github Repository of the framework. Available on: `https://github.com/syri0/cmscan/tree/master/data`, [Online; Last accessed: 31 August 2019]
2. http-drupal-enum Script. Available on: `https://svn.nmap.org/nmap/scripts/http-drupal-enum.nse`, [Online; Last accessed: 04 September 2019]
3. http-wordpress-enum Script. Available on: `https://svn.nmap.org/nmap/scripts/http-wordpress-enum.nse`, [Online; Last accessed: 04 September 2019]
4. List of data in NSE Libraries. Available on: `https://svn.nmap.org/nmap/nselib/data/`, [Online; Last accessed: 04 September 2019]

5. Market Share:Top Website Platforms and Example Sites. Available on: `https://websitesetup.org/popular-cms/`, [Online; Last accessed: 29 November 2017]
6. Nessus Vulnerability Scanner. Available on: `https://www.tenable.com/products/nessus-vulnerability-scanner`, [Online; Last accessed: 29 November 2017]
7. Nmap-Network Mapper. Available on: `https://nmap.org/`, [Online; Last accessed: 29 November 2017]
8. NSE-Nmap Scripting Engine. Available on: `https://nmap.org/book/nse.html`, [Online; Last accessed: 29 November 2017]
9. SAINT Cybersecurity solution. Available on: `http://www.saintcorporation.com/`, [Online; Last accessed: 29 November 2017]
10. Security Auditor's Research Assistant. Available on: `http://www-arc.com/sara/`, [Online; Last accessed: 29 November 2017]
11. VLAD the scanner. Available on: `http://www.decuslib.com/decus/vmslt00b/net/vlad_readme.html`, [Online; Last accessed: 29 November 2017]
12. Website Hcked Trend Report. Available on: `https://sucuri.net/website-security/website-hacked-report.`, [Online; Last accessed: 29 November 2017]
13. 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)
14. 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)
15. 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)
16. 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)
17. Meike, M., Sametinger, J., Wiesauer, A.: Security in open source web content management systems. IEEE Security & Privacy **7**(4) (2009)
18. Scott, D., Sharp, R.: Developing secure web applications. IEEE Internet Computing **6**(6), 38–45 (2002)
19. 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)
20. 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)