IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 12, 2019, accepted December 2, 2019, date of publication December 4, 2019,
date of current version December 18, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2957573

An Intelligent and Automated WCMS Vulnerability-Discovery Tool: The Current State of the Web

PRIMOZ CIGOJ[1,2] **AND BORKA JERMAN BLAZIC**[2]
[1]Jozef Stefan International Postgraduate School, 1000 Ljubljana, Slovenia
[2]Laboratory for Open Systems and Networks, Jozef Stefan Institute, 1000 Ljubljana, Slovenia

Corresponding author: Primoz Cigoj (primoz@e5.ijs.si)

This work was supported by the Slovenian Research Agency under Grant 34610.

ABSTRACT The main focus of the cyber-security community has been to make operating systems and communication networks more secure and harder for attackers to penetrate. This also applies to all the applications on the Internet. The most frequently used web application and user pages are today developed with the Web Content Management System (WCMS), as it allows user-friendly access, easy development and operation. WCMSs are present all over the world in many different environments. Any malware that can penetrate the WCMS can significantly affect the system itself and can cause a misconfiguration. The security and stability of these systems are important for reducing the risks and consequences of attacks or disfunctions with currently known similar tools. This paper presents a newly developed tool for identifying the vulnerabilities of the majority of Internet sites with WCMS applications and the remedies to be applied. Its key feature is the ability to perform automated, fast and dynamic vulnerability scans of the WCMS and the attached plug-ins on a large scale with in-built ethical respect.

INDEX TERMS Cyberspace, databases, software, security, search engines, search methods, metasearch, web search, websites, WCMS.

I. INTRODUCTION

We do business, pay through the Internet, store documents and share our personal information, card numbers and identification numbers online very frequently. There is no doubt that this data is private and should be stored as safely as possible. The vision of the future Internet involves building a new generation of applications made by merging services and data from different providers and organizations [1]. Web services provide the basic interface between the provider and the consumer, supported by complex software component like the operating system, the server application and many additional systems like databases, shops and selling systems, appliances for different services, etc. Web services are subject to several unique security concerns, due to their pervasiveness, seamless interoperability and operations that can be remotely invoked by the user, and they need to be carefully considered in view of the envisioned architecture of the future Internet. The major web-security concern is related to the differences between the web applications that do not have embedded security protection and the security solutions present in traditional messaging techniques applied within other Internet services. For instance, the SOAP protocol used in web-service communications does not address the security itself and can be bypassed by a firewall. Initiatives to improve web-service security have been proposed [2]. During the last Europol C3 workshop at the "The International Cyber Security Summer School (ICSSS)", experts suggested that a fast, reliable and applicable tool for searching web vulnerabilities across the whole internet is needed to improve the overall security for the provision of a secure future Internet.

This article presents a newly developed tool called VulNet for measuring the vulnerability of all the Internet's websites with WCMS (Web Content Management System) applications that overcomes the deficiencies associated with similar tools. The tool is an advance over other known proprietary or open-access tools as it can scan all the Internet websites with WordPress WCMS in an acceptable time frame, with a speed for scanning and identifying the vulnerabilities

The associate editor coordinating the review of this manuscript and approving it for publication was Leandros Maglaras.

TABLE 1. Properties of known web-scanning tools.

Characteristics / Tools		Shodan	Censys	WPScan	[5]	[11]	[12]	[13]	VulNET
General Characteristics	OpenSource	No	No	Yes	No	No	No	No	Yes (A)
	URL or IP	IP	IP	URL	IP	URL	URL	URL	Both
	Results are Freely Accessible on the Internet	Yes (P)	Yes	No	No	No	No	No	Yes
	Internet-connected Devices	Yes	Yes	No	Yes	No	No	No	Yes (E)
	Automatic Scanning	Yes	Yes	No	Yes	Yes	No	No	Yes
	Ethical	Yes	Yes	Yes (P)	Yes	No	Yes	Yes	Yes
	Web UI and Command Line (CL)	Both	Both	CL	CL	No	No	No	Both
	Real-time Visualization while Scanning	No	No	No	No	No	No	No	Yes
	Free API	Yes (P)	No	No	No	No	No	No	Yes
	More than 1 million scanned IPs or Websites	Yes	Yes	No	Yes	No	No	No	Yes
WP Specific	CMS Scan	No	No	Yes	Yes	Yes	Yes	Yes	Yes
	CVE Exposer	Yes	No	Yes	Yes	No	No	No	Yes
	Plugins Scan	No	No	Yes	No	No	No	Yes (L)	Yes
	Scoring	No	No	No	No	No	No	No	Yes

P = Partly, E = Extension, A = Attended, L = Limited.

of the websites and the attached plug-ins that is several times faster than the existing solutions.

At the same time the tool provides an assessment of the threatened security using an original developed scoring system for the detected vulnerabilities. The main value of the tool is in its ethical type of scanning applied by respecting the readiness of the website to allow browsing of the WCMS pages and the attached plug-ins. This property is not present in most of the known web scanners. The innovative aspects of the tool and its capabilities are compared with known scanners and tools and presented in Table 1. The VulNet tool offers a more accurate and intelligent detection of the vulnerabilities. When scanning the Internet's web servers VulNet identifies the servers hosting several WCMS web pages (e.g., known as a virtual web server) and analyses and assesses the vulnerabilities in all the present website pages and within the implemented plug-ins. The property is enabled with an original in-built algorithm presented in Appendix 1.

However, it should be noted that the focus is on the results found by reviewing the whole web system and how the security picture changes due to the detected and measured vulnerability. Its key feature is an ability to perform an automated, fast and dynamic vulnerability scan of the WCMS and attached plug-ins on a large scale with in-built ethical respect.

The paper is divided into seven sections. After the introduction, the second section introduces the reader to the area and describes the problem being addressed. The next section provides information about previous work in the area of web-vulnerability detection, while section four describes the applied methodology and the tool components. The fifth section discusses the results. In the next section the limitations of the tool are elaborated together with its usability. The paper ends with a conclusion and a discussion, where the potential international service development for informing the owner of the website about the detected vulnerabilities is briefly presented.

II. OVERVIEW OF THE AREA AND THE PROBLEM

The continuous evolution of networks based on Internet technology has made its services very attractive and many different new applications appeared with use of WCMS. A modern content-management system (CMS) like WordPress simplifies website creation as it allows the functionality of the site to be extended with additional applications known as plug-ins that are available for downloading from known databases. Currently, the estimated number of plug-ins is close to 54,000 and the total number of downloads is close to 900 million. Public web applications are usually accessible from anywhere in the world, but many corporate web applications that are set on networks with restricted access are also accessible. Web applications handle very sensitive information, ranging from banking to health directories, as well as personal images and photographs that are of interest to criminals and attackers. Even applications that do not handle sensitive or valuable data have become interesting targets, as they can be used for various unlawful purposes, such as serving malicious content or exploiting knowledge for nefarious purposes, motivated by economic, political or religious reasons. According to a survey carried out by W3Tech, about 52.9% of Internet websites use some kind of web-content management system [3]. The most popular open-source web-content management systems (WCMSs) are WordPress, Drupal and Joomla [4]. A technology survey by BuiltWith Pty Ltd in 2017 concluded that about 46% of the top one million websites use WordPress. The reason is that WCMSs allow users, even without an in-depth knowledge of web technology, to deploy and offer system content to users. Due to the popularity of these systems, they have become an interesting target for malicious attackers, and therefore the importance of the security features and the overall vulnerability of these applications have become very important, especially in cases where the web-content owners do not possess the necessary knowledge and understanding of the possible threats to the system. Writing computer programs is a complex task and modern software-development usually involves combining

many libraries and frequently not all the bugs have been removed. Design errors become a risk, especially when the security of the program has not been taken into consideration from the beginning of the design process. The architecture and the design of a computer system are expected to be coherent and to follow the security principles, but this is not the case in the current web space [5]. Knowing the system's vulnerability represents the most vital and precious information for malicious parties. The removal of the vulnerability increases the resilience of the underlying system. That is why the operation and management of the web system should be actively monitoring and removing the vulnerable parts of the system to prevent possible attacks. There are many types of web attacks that can trigger very different outcomes, from a misfunction of the managed system through data theft and its misuse. The attacks could be at the database level, such as SQL injection, or at the web-server level, in an attempt to manipulate web parameters, upload malicious files, bypass user authentication, intrude on the system, elevate privileges, install a spam relay or hijack a session. A particular web user can also become a victim, with an attack known as cross-site scripting, that appears in the form of a browser-side script that redirects the user to a fake site (phishing) or stealing information like credit-card numbers. Information about these attacks, with potential successful outcomes of the site's vulnerability exploited, is precious information if it is passed to the development or maintenance teams so that the system's security can be further improved and the models for preventing threats are updated. Monitoring third-party libraries or providing information about the existing vulnerabilities of plug-ins and the available patches to remove them is also necessary for all operating systems. The vulnerability testing of websites can be performed using two approaches. One is called white-box testing, in which the testing software has access to the source code of the application and this source code is then analysed to track down defections and vulnerabilities in the code. These operations are expected to be integrated into the web-development process with the help of add-on tools within the development environments, but they are usually not used, especially when the system is upgraded with new plug-ins to enhance the service and user satisfaction. The other approach is called black-box testing, where the tool has no direct access to the source code, but instead tries to find vulnerabilities and bugs with special input test cases that are generated and then sent to the application. Responses are then analysed for unexpected system behaviours that indicate the errors or vulnerabilities of the system. A black-box security scanner typically uses a mixture of passive (typically, during the crawl) and active (typically, post-crawl) vulnerability-testing techniques like code execution [6]. Identifying the vulnerabilities and their measurement across the whole web space of the Internet is not an easy task, though this information is extremely valuable, helpful and required by the website owners. Measuring the vulnerability and the provision of the appropriate patches ensures the overall Internet security and the website's performance. The available vulnerability-testing tools are either restricted to internal use by the owners of corporate or organization networks, as they use software mimicking real attacks, or they just scan the basic web server's vulnerability, without providing sufficient information about the whole WCMS system and the associated plug-ins. In the continuation of the paper a newly developed tool that provides a very efficient measure of the vulnerability of all the Internet sites with WCMS applications that overcomes the deficiencies of the known tools are described, but the focus is given to the results found from reviewing the whole system and how the overall security picture changes due to the detected and measured vulnerability.

III. OVERVIEW OF THE PREVIOUS WORK

A common approach to measuring web vulnerability is scanning the Internet sites and associated domains with the use of a web crawler in combination with a search engine, such as Google. Web crawlers, however, have multiple problems. Some crawlers access the same URLs [7] more than 1000 times, as there is no intelligence in-built into the crawler. Any web-vulnerability measurement of the web application, besides the crawler, needs additional software, as the information sought beyond the port data is located in the plug-ins and the application itself. The detection of infinite loops and the actual depth of crawling in the web space are difficult as the websites are not static and the web pages change over time due to user intervention. A more difficult problem is related to the large number of pages and the amount of data included. As a consequence, the crawling process can take a long time and the results are frequently not a snapshot of the system, as multiple pages might have changed during the scan. However, in recent years some improvement to Internet-wide scanning was achieved with tools such as ZMap and MasScan [8]. ZMap was developed by the University of Michigan and is now the main tool of the Internet-search service known as Censys [9]. Shodan is a similar service that uses behaviour or grab techniques to identify the vulnerabilities of sensors and similar devices. Shodan collects data mostly on web servers, but it is supplied with applications to access FTP servers and other known Internet ports such a Telnet (virtual terminal), SNMP (mail), IMAP (encrypted mail) and the Real Time Streaming Protocol. The latter are used to access web cameras and their video stream. However, Shodan does not conduct a deep review of the sites. Despite the popularity of these tools, they are not real crawlers, but rather ports scanners looking for the HTTP type of servers that are usually extended with additional applications. The collected port information using these tools does not include information about the vulnerability of the applications and the plug-ins as they operate only with an IP address and do not crawl links within the website's content. The systems are proprietary, but the service is publicly available. A similar publicly available tool is called Nmap [8], which requires multiple machines and weeks to complete any horizontal scan of the public address space, making it rather slow [5]. Running regular web vulnerability scanners against

numerous websites is time consuming and, if exploit techniques are used, the scan is considered as illegal if browsing permission is not granted by the owners. Another way of detecting the vulnerability without breaking the law [6] is to detect the application and then identify its issuing version or its fingerprint and then look in a database with the identified vulnerabilities of that particular version. The most well-known vulnerability database is the National Vulnerability Database (NVD) that is hosted by the National Institute of Standards and Technology [10]. The NVD includes several databases, ranging from security checklists to security-impact metrics.

Several researchers have created search engines that make it possible to determine whether the website is vulnerable; nevertheless, each of the created engines faces a lack of features in order to cover all the necessary aspects. We could split them into two groups. The first group uses data sets between 20,000 and 200,000 websites [11]–[13], but forgetting the rest of the web, and they are focusing on specific vulnerability, such as XSS, SSL, SQL injection, phishing, Heartbleed and search-redirection attacks, instead of covering all of them at once. In addition, their methods are also time-consuming: they need more than 9 days to measure a dataset of 200,000 websites. They focus on determining whether a given input propagates, rather than efficiently finding the propagating inputs, for arbitrary vulnerabilities [12], [14]. The second group performs the scanning of the Internet ipv4 protocol for a specifically defined subject area, such as hosted services, SSL/TLS, vulnerabilities or specific software or protocol vulnerabilities by using mass scan tools such as ZMAp, Nmap and Massscan [15]–[17]. This technique is good for the fast TCP/IP stack-fingerprinting technique to identify the OS's type, port range scan, and basic web, but not for a detailed overview of the online vulnerabilities. In this case they are forgetting about the CMS's core and plug-in vulnerabilities.

Table 1 presents the results obtained from the properties comparison of the best-known vulnerability scanners and the VulNet tool. The two proprietary tools, Shodan and Censys, are scanning web servers only, as their main task is to detect the vulnerability in web-connected devices like cameras, sensors, etc. in the IoT environment. The other listed scanners detect the vulnerabilities of WCMS and show some common properties with the VulNet tool; however, none of them contains all the listed tool characteristics presented in Table 1.

The search for vulnerabilities that focuses on the content applications was made in recent years by several authors from academia. Their attempts, unfortunately, were applied to a relatively small sample of web servers. The reason for that lies in the difficulties in recognizing direct, potentially malicious, security breaches and in the fact that the scanning of the whole Internet WordPress sites, due to the required time for scanning, was almost impossible. However, even these studies on a small number of samples found that the WordPress servers are not fully protected and that they are exploitable in most cases due to the use of third-party plug-ins. One of the studies discovered the presence of 860 vulnerabilities in a set of 127 plug-ins, belonging only to 322 WCMS sites. Sensitive directory information like in health systems was disclosed during one of these studies [18]. Most of these tools were developed for specific exploits, such as SQL injection and cross-site scripting vulnerability. The detection was also based on a symbolic code execution for PHP applications [19]. In 2014 a tool known as Wasapy was developed and was used as a black-box scanner [20]. The search for vulnerability with Wasapy was focused on detecting the code-injection vulnerability. The research interest of these studies was guided mainly by the popularity of WCMS, as most of them were focused on only one or two different vulnerabilities. From that finding it follows that although various vulnerability-testing systems exist, either on the market or as freely available software, including the academic attempts, a fully automated tool for a dynamic vulnerability measurement of a WCMS across the whole Internet is not yet available.

The following describes an attempt to fill this gap and to provide a real measure of the Internet's web vulnerability with the VulNet tool. The identification of the website's vulnerability and that of the attached plug-ins, i.e., a clear image of Internet security holes, can produced and solutions can be more quickly applied by the web-production teams that are usually not aware of the potential threats. However, an automated notification of them needs to be applied. The major contribution of this paper is a tool that will allow an automated, very fast, dynamic vulnerability scan of a WCMS and the attached plug-ins on a large scale with in-built ethical respect.

IV. THE APPLIED METHODOLOGY AND THE TOOL

The general methodology for web scanning and data collecting consists of six steps: a) collecting the IP addresses of the web targets, b) accessing and c) getting responses, d) sending queries for application patterns, e) collecting vulnerability information and f) saving and validating the results. The last three steps in current web scanners differ very much due to their capability to catch relevant objects, the range of the collected data, the speed of the provision of answers and the vulnerability analysis provided. The VulNet tool offers an effective search for vulnerable servers on a large scale, so it scans the whole web rather than just the IPv4 addresses, like most other tools (Zmap, nmap, massscan, etc.). Admittedly, Zmap can scan a huge number of IPv4 addresses, but access to the IPv4 address gives us direct access to only one website hosted on this ipv4 address and not the rest that could be located at this same IP. A web server can be configured to serve multiple virtual websites (in some cases more than 1000 websites) from a single IPv4 address. The VulNet tool presented below offers a more accurate vulnerability measurement by implementing innovative solutions in the scan part of the tool and in the vulnerability identification and analysis.

The methodology that was followed in designing and implementing the tool is known as a software-design process that envisions and defines the software solutions for one or more sets of problems. The main component of every software design is the software-requirements analysis, which lists the specifications that are then used in the programming with the methods of software engineering. There are many known models for software design and engineering; however, the whole process and algorithm generation that are usually a final result of the methodology applied are presented with a pseudocode. In the case of the VulNet tool the pseudocode of the vulnerabilities' analyser is presented in Appendix 2.

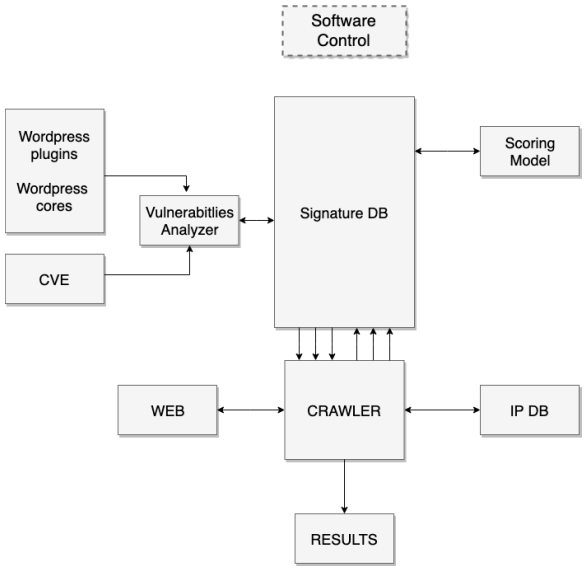

FIGURE 1. Components of the VulNet tool.

The VulNet tool is built from four modules presented on Figure 1. The first module is the local Signature database, which stores all the known WordPress plug-ins and different core versions of the WCMS. The second module is the Common Vulnerabilities and Exposures (CVE) database, built up from different public resources available online (CVE databases, Exploit databases, etc.). In the production of these two blocks a specific methodology for scoring the vulnerability was used that indexes each known plug-in or version of the WCMS. The value of a particular index depends on the assessment of the potential threat if the vulnerability is exploited. The CVE database contains 300 identified vulnerabilities of the WordPress core versions and more than 1300 WordPress plug-ins exploits. The index value is assigned based on the developed scoring mechanism that enables easier verification of the potential damage and the vulnerability assessment. The index values for seven known vulnerabilities run from 3 to 9, but the group of all the vulnerability types is scored to 10.

The innovative aspect of the VulNet tool is in the implementation of the innovative searching and scoring mechanisms. The implemented search mechanism of the tool is fast and reduces the likelihood of false access due to the feedback information received from the crawler and the matching with the information in the temporary database that prevents repeating access to already-visited servers. The specific scoring mechanism enables a rapid and reliable vulnerability analysis and assessment. Since there is no list of WordPress sites on the Internet, the first step in the search process is to scan the entire web space and to find the sites with a WCMS. According to Verisign's data (verysign.com) the web space is enormous, as there are more than 350 million web domains registered on the Internet. Unfortunately, not all DNS (Domain Name Server) zones are accessible (due to private networks) and for that reason a list of root domains is created in the initiation operational phase of the tool. To speed up the process, all the services that allow the use of shortened URLs, and large sites like Facebook, YouTube, and Instagram are removed from the search list as they are not operating any significant device. After identifying the presence of a WCMS the query part of the tool looks for the presence of the file under the name "robots.txt", used to verify whether the site allows browsing, meaning that the reviewing of applications is legal. The next step is sending queries to the site and collecting information about the content.

The tool looks for the web meta tag generator (<meta name ="generator" content ="WordPress 5.1.1">), which usually contains information about the version of the platform. If the data is missing, then the tool looks for CSS and JS files, (/wp-includes/js/wp-emoji-release.min.js?ver = 5.2); this information is provided at the end of the file. The tool then parses the plug-ins (wp-content/plugins/wp-hide-post/public/js/wp-hide-post-public.js?ver = 2.0.10) to obtain the version applied. If the version of the plug-ins is not available in the CVE database, then the plug-ins are stored in a separate folder for further analysis and a search of exploits. The server type (e.g.Apache/2.2.15 CentOS), and the IP address are also stored and used later to identify the server's location and the country of origin. The scoring assessment for the vulnerability of the website's page is based on two sets of parameters: the first set is used for the risk assessment of the website's core $C_{(s)}$, and the second set is used for a risk assessment of the attached plug-ins $P_{(s)}$. The version of the website's core is first verified and then the tool looks for the potential vulnerability of the core in the signature database. In the case of several identified vulnerabilities, the score with the highest value is selected for the website core's risk parameter. The same approach is applied to the plug-ins: the first match is found for each of the plug-ins and then the vulnerability with the highest risk parameter is selected for the plug-in's risk value.

The calculated vulnerability-risk score for the web WCMS $A_{(s)}$ is calculated as an aggregated score from the score obtained for the web-server analysis of the website's core $C_{(s)}$ and the score for the attached plug-ins $P_{max(s)}$:

$$A_{(s)} = \frac{C_{(s)} + P_{max(s)}}{2} \tag{1}$$

where $A_{(s)}$ is the final risk score, $P_{max(s)}$ is the largest parameter among the parameters for the detected plug-ins'

vulnerabilities and $C_{(s)}$ is the highest score among the vulnerabilities detected for the web server as well.

The scan speed and the answers that provide the data are very important characteristics of any scanning tool. VulNet is capable of scanning 90,000 web pages in 15 minutes, which means that the results for 6000 websites can be viewed in less than a minute. This classifies the tool as very fast compared to existing tools. The answering speed is variable, as the response of the WCMS pages depends on how fast the web server is at delivering the responses. Some servers need up to 15 seconds to respond and that timing influences the speed of the data collection. The program's logic is written in the Python programming language.

V. RESULTS

Several experiments were performed with the tool and the results are very promising. It seems that a relevant service for end users and the WCMS's developers can be provided. The scanning results provided by the VulNet tool are displayed with a web interface developed with PHP. The interface displays live scanning and the obtained data. It also maps the path of the crawler around the map. A screenshot of the running portal with the crawler path and the reviewed WCMSs with their locations are shown in Fig. 2.

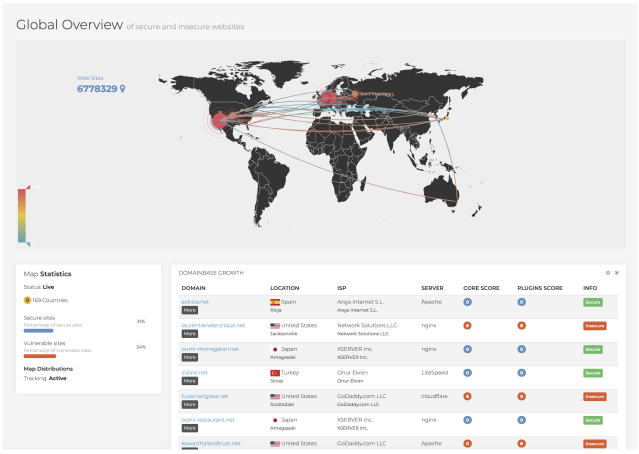

FIGURE 2. Real-time monitoring platform.

Fig. 2 shows the results for the data collected from 115 million randomly scanned web addresses from around the world (over 194 countries). The tool established 126,086,633 links; however, it should be noted that some web servers were not active or exceeded the waiting time of 15 seconds. Among these sites there were 16,274,980 valid WordPress installations and 14,887,047 plug-in installations. In the identified web set more than 5,018,262 were found to be vulnerable, representing 31% of all the WordPress installations on the Internet, where 2,475,337 had a higher score than 5. A total of 4,356,067 vulnerabilities were detected among the detected plug-ins and 2,795,855 had a score higher than 5. The analysis of the results revealed that there are very vulnerable core versions of WordPress, recent versions of WordPress (4.9.8, 4.9.3, 5.0.3, 4.9.6, 4.9.7, 4.9.4, 5.1, 4.9.5, 4.6.1), accounting for over 1 million vulnerable pages.

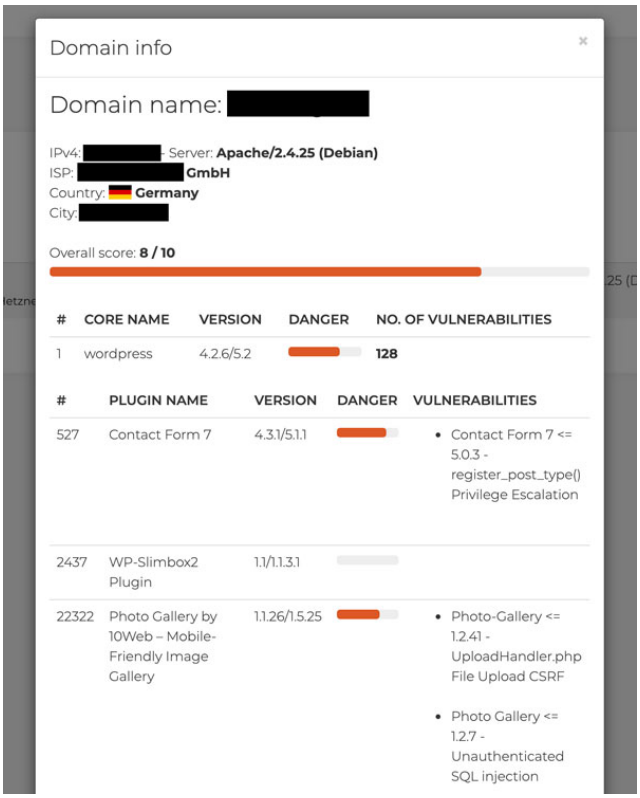

FIGURE 3. The measured vulnerability of the affected domain.

The United States, Germany, France, Netherlands, and United Kingdom were the top 5 countries with the most vulnerable plug-ins and core versions. The measured vulnerability of the affected domain (e.g., the country top-level domain) is shown in Fig. 3.

VI. LIMITATION AND USABILITY OF THE TOOL

The VulNet tool was developed to offer Internet WordPress website owners and users easy access to the data providing an image of the Internet web vulnerability. In that context the usability of the tool is important. The VulNet user interface is built with web technology and is very simple to use. After accessing the user-friendly client, the user enters the site domain or IP address and this triggers the scan and the display of the results. The user interface and the user facilities are illustrated in Fig 2. and Fig 3. The tool also enables a selective search depending on the user's needs and requirements. This feature makes it possible to detect the vulnerabilities within a particular web server, identified by its IP or domain name. Additionally, the selection allows the user to ask for automatic feeding of information in the future about his/her website's vulnerabilities in a selected period of time. This service is important as current exhaustive studies for setting a central awareness service for informing the web owner and the users about the identified vulnerabilities have shown that such a service would not be successful, either due to user ignorance about the mailed information or due to a reluctance to learn about security or simply because they do not trust

an unknown service [12]. The VulNet option is open to the user's needs, which seems to be especially important for web-hosting providers.

Like any novel tool and approach, VulNet shows some minor limitations in implementing the major goal - detecting vulnerable websites. VulNet's detection of vulnerabilities is only focused on the WordPress web servers and the associated plug-ins. The decision to develop a tool focused on WordPress websites' applications is because of WordPress's large presence on the Internet, which is due to its popularity and ease of use and system maintenance. WordPress applications and the developed plug-ins for implementing different services are free of charge and they provide an easy setting for various services and miscellaneous applications. These properties have led to WordPress being popular with not-for-profit organizations and with owners that do not have regular professional system-maintenance services. In addition, most of these owners were not sufficiently aware of the missing security support to the web server and the applications that protect the web operations, the client and the data [21]. As a consequence, as was shown in previous studies, the vulnerabilities of WordPress websites are widespread on the web.

VII. CONCLUSION AND DISCUSSION

The current known services and tools for measuring the WCMS vulnerabilities lack to provide real insight in the security of the websites operating with WordPress applications [22]. They usually provide only raw data, the address of the accessed server and the associated ports. The effectiveness of these known security tools is low as they do not provide information about the security holes within the web applications supported by the installed plug-ins. WCMS owners need vulnerability information that involves the multiple, attacker-controlled dimensions of the possible exploits. The advantage of the VulNet tool is in the scanned search space in which almost all the known vulnerabilities and potential exploits can be found. The advantage is in the applied scanning approach that allows a fast and reliable overview of almost all the accessible web space. By identifying the vulnerability, safe patch fingerprinting based on known and recently identified signatures can be applied and the vulnerability removed.

An indication that a vulnerability measurement of WCMS improves the overall web security is the changed situation in the web space after May 2019. The number of vulnerable WordPress site installations fell sharply after May 2019, when the WordPress organization released for the first time since the first release of the WordPress application in 2007, an update of the core releases (core versions 1, 2, 3 and 4). After rolling out information about the dimension of the vulnerabilities in the Internet WBCMS space the organization took care of all previously not-updated versions of the WordPress applications with solutions that removed the identified vulnerabilities. After May 2019 the number of vulnerable sites on the Internet WBCMS fell by 50% compared to the number of vulnerable sites before May 2019.

APPENDIX 1 - IN-BUILT SCANNING ALGORITHM

```
procedure CrawlerThread(queue)
    while queue > 0 do
        website <- queue.nextURL()
        url <- website.URL()
        if website.permitsCrawl() then
            content < retrieveURL(url)

            // We store Domain info
            // IPv4, ASINFO, City, CountryCode, ISP, Latitude, Longitude,
            // Organization, Region, TimeZone, ZIP number, Server Type

            storeWebsite(content, url)

            for each url in parse(content) do

                // We find domain neighbours on the same server address
                neighbours <- url.getNeighbours()

                if neighbours > 0 then
                    for each n in neighbours do
                        // We add new domains into the queue
                        if n not in queue and n not like socialLinks():
                            queue.addURL(n)
                        end if
                    end for
                end if

                // We add new domains into the queue
                if url not in queue and url not like socialLinks():
                    queue.addURL(url)
                end if
            end for

            // We do not want to verify domain which is not Wordpress
            if website.wordpress() then
                // We check Wordpress Core and Plugins vulnerabilities
                website.identifyVulnerabilites(content, url)
            else
                website.addToIgnoreList()
            end if
        end if
        queue.releaseWebsite(website)
    end while
end procedure
```

APPENDIX 2 - ALGORITHM FOR MATCHING AND CHECKING CORE OR PLUGINS VULNERABILITIES

```
procedure identifyVulnerabilites(content, url)

    // We already know its Wordpress - verified before
    // First we check Wordpress Core Version

    domainID <- url.getDomainID()
    wordpress_version <- parseCoreVersion(content, domainID)

    // checkCoreVuln verifies in our Signature Database
    // vulnerabilites for Wordpress Core Version.

    if wordpress_version then

        // we retrive signatureID record with identified vulnerabilites
        c_secure, c_score, signatureID <- checkVuln('core', wordpress_version)

        // We store in our database Wordpress core version and score
        storeCore(wordpress_version, signatureID, c_secure, c_score, domainID)

    end if

    if hasPlugins(content) then

        max_score <- 0

        // We parse all the plugins available on the site
        for each plugin in parsePlugins(content)

            p_version <- plugin.Version()

            p_secure, p_score, signatureID <- checkVuln(plugin, p_version)

            // We store in our database Plugin version and score
            storePlugin(plugin, signatureID, p_secure, p_score, domainID)

            if max_score < p_score:
                max_score <- p_score
            end if
        end for
    end if

end procedure

procedure checkVuln(slug, slugVersion)

    // slug <- 'wysija-newsletters'
    // slugSearch <- 'wysija +newsletters'

    // We try to find match between parsed plugin and plugin in our database

    procedure calculateMatch(slug, signature)
        return 1 / (abs(length(slug) - length(signature.slug)) +1)
    end procedure

    for (signature, version) in database do
        if version.version  equals slugVersion than
            match <- 0
            if signature.slug equals slugSearch then
                match <- calculateMatch(slug, signature)
            end if
            results.add([version, signature, match])
        end if
    end for
end procedure
```

REFERENCES

[1] S. Karumanchi and A. C. Squicciarini, "A large scale study of Web service vulnerabilities," *J. Internet Service Inf. Secure*, vol. 5, no. 1, pp. 53–69, 2015.

[2] The Hague Security Delta. (Oct. 2018). *The Hague Security Delta*. Accessed: May 2018. [Online]. Available: https://www.thehague securitydelta.com/news/newsitem/976

[3] *W3Techs*. Accessed: May 2019. [Online]. Available: https://w3techs.com

[4] M. Hassan, K. Sarker, S. Biswas, and H. Sharif, "Detection of wordpress content injection vulnerability," 2017, *arXiv:1711.02447*. [Online]. Available: https://arxiv.org/abs/1711.02447

[5] P. Laitinen, "Vulnerabilities in the wild: Detecting vulnerable Web applications at scale," M.S. thesis, Univ. Jyväskylä, Dept. Comput. Sci. Inf. Syst., Jyväskylä, Finland, 2018.

[6] H. Trunde and E. Weippl, "WordPress security: An analysis based on publicly available exploits," in *Proc. 17th Int. Conf. Inf. Integr. Web-Based Appl. Services*, vol. 81, 2015, pp. 1–7

[7] *Uniform Resource Location—Identification of the Web Site by Its Internet Address (IP)*. Accessed: May 2019. [Online]. Available: https://www.w3.org

[8] A. Tundis, W. Mazurczyk, and M. MÜhlhäuser, "A review of network vulnerabilities scanning tools: Types, capabilities and functioning," in *Proc. 13th Int. Conf. Availability, Rel. Secur.*, vol. 65, 2018, pp. 1–10

[9] Z. Durumeric, E. Wustrow, and A. J. Halderma, "ZMap: Fast Internet-wide scanning and its security applications," presented at the 22nd USENIX Secur. Symp. (USENIX Secur.), 2013.

[10] NIST. *National Vulnerability Database*. Accessed: May 2019. [Online]. Available: https://nvd.nist.gov

[11] T. V. Goethem, P. Chen, N. Nikiforakis, L. Desmet, and W. Joosen, "Large-scale security analysis of the Web: Challenges and findings," in *Proc. Int. Conf. Trust Trustworthy Comput.* New York, NY, USA: Springer-Verlag, 2014, pp. 110–126.

[12] B. Stock, G. Pellegrino, F. Li, M. Backes, and C. Rossow, "Didn't you hear me?—Towards more successful Web vulnerability notifications," in *Proc. Netw. Distrib. Syst. Secur. (NDSS) Symp.*, 2018.

[13] M. Vasek, J. Wadleigh, and T. Moore, "Hacking is not random: A case-control study of Webserver-compromise risk," *IEEE Trans. Dependable Secure Comput.*, vol. 13, no. 2, pp. 206–219, Mar./Apr. 2015.

[14] N. Schagen, K. Koning, H. Bos, and C. Giuffrida, "Towards automated vulnerability scanning of network servers," in *Proc. 11th Eur. Workshop Syst. Secur.*, vol. 5, 2018, pp. 1–6

[15] A. Nappa, Z. M. Rafique, J. Caballero, and G. Gu, "CyberProbe: Towards Internet-scale active detection of malicious servers," in *Proc. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, 2014, pp. 1–15.

[16] H. Kim, T. Kim, and D. Jang, "An intelligent improvement of Internet-wide scan engine for fast discovery of vulnerable IoT devices," *Symmetry*, vol. 10, no. 5, pp. 151–166, 2018.

[17] F. Li, Z. Durumeric, J. Czyz, M. Karami, D. McCoy, S. Savage, and V. Paxson, "You've got vulnerability: Exploring effective vulnerability notifications," in *Proc. 25th USENIX Secur. Symp. (USENIX Secur.*, 2016, pp. 1033–1050.

[18] T. Koskinen, P. Ihantola, and V. Karavirta, "Quality of WordPress plug-ins: An overview of security and user ratings," in *Proc. Int. Conf. Privacy, Secur., Risk Trust Int. Conf. Social Comput.*, Sep. 2012, pp. 834–837.

[19] G. Agosta, A. Barenghi, A. Parata, and G. Pelosi, "Automated security analysis of dynamic Web applications through symbolic code execution," in *Proc. 9th Int. Conf. Inf. Technol.-New Gener.*, Apr. 2012, pp. 189–194.

[20] R. Akrout, E. Alata, M. Kaaniche, and V. Nicomette, "An automated black box approach for Web vulnerability identification and attack scenario generation," *J. Brazilian Comput. Soc.*, vol. 20, no. 1, p. 4, 2014.

[21] M. Hassan, K. Sarker, S. Biswas, and M. Sharif, "Detection of Wordpress content injection vulnerability," 2017, *arXiv:1711.02447*. [Online]. Available: https://arxiv.org/abs/1711.02447

[22] I. C. Cernica, N. Popescu, and B. Tiganoaia, "Security evaluation of wordpress backup plugins," in *Proc. 22nd Int. Conf. Control Syst. Comput. Sci. (CSCS)*, May 2019, pp. 312–316.

PRIMOZ CIGOJ is currently with the Jozef Stefan Institute, Ljubljana, Slovenia. Besides that, he is completing the doctoral dissertation in the information and communication technologies with the Jozef Stefan International Postgraduate School. He participates in European-funded H2020 projects. He is an Ethical Hacker and has over 15 years of experience carrying out security checks and penetration tests. His main areas of interests are information security, digital forensics, fight against cybercrime, and cloud computing. Moreover, he has been the Internet Society (ISOC) Representative in Slovenia, since 2011.

BORKA JERMAN BLAZIC is currently a Full Professor with the Department of Economics, University of Ljubljana, and is heading the Laboratory for Open Systems and Networks, Jozef Stefan Institute. At the University of Ljubljana, she is teaching courses in electronic communication and information security. She is also a Professor with the Postgraduate International School Jožef Stefan, teaching courses on internet technology. She has been employed with the Unit of Computer Security, Stockholm University, from 2011 to 2015. She spent her postdoctoral study with the Iowa State University, Ames, IA, USA, and has worked as a Project Development Officer for TERENA - The European Association of Academic and Research Networks, Amsterdam. She was leading research teams in many EU.