Plugins to Detect Vulnerable Plugins: An Empirical Assessment of the Security Scanner Plugins for WordPress

Daniel T. Murphy
University of New Orleans, USA
dtmurph1@uno.edu

Minhaz F. Zibran
Idaho State University, USA
MinhazZibran@isu.edu

Farjana Z. Eishita
Idaho State University, USA
FarjanaEishita@isu.edu

***Abstract*—WordPress, possibly world's the most popular Content Management System (CMS), which supports around 455 million websites and claims 60.3% of all content management systems in use. The WordPress core is known to be relatively secure, but its plugin ecosystem is not. 92% of vulnerabilities found in WordPress powered websites are attributed to third-party plugins that those websites depend on.**

This paper presents an empirical study, where we examine the efficacy of 11 WordPress security scanner plugins in the detection of known vulnerabilities in another set of 51 insecure plugins. The results are mixed, with some security scanner plugins failing entirely and even the most effective plugins failing to identify significant vulnerabilities. The findings are derived based on both a quantitative analysis and a deeper qualitative analysis.

***Index Terms*—Security, Vulnerability, WordPress, Plugin, Website, Web App, Empirical Study**

I. Introduction

Web applications and web services are ubiquitous today, and their use is ever increasing with the advancement of science, technologies, and businesses. This growing availability of web applications and web services presents attractive target for malicious cyber attacks.

To facilitate the growing demand of web applications and web services, a number of content management systems (CMS) have become available. Such content management systems allow users to easily create and deploy websites and web applications, often requiring minimal or no technical knowledge to use. Thus, many individuals as well as small to medium businesses launch websites powered by CMS.

WordPress is possibly the most used content management system by a large margin, mainly functioning as the back-end for hundreds of millions of websites and claiming 60.3% of all content management systems in use [1], [24]. WordPress is available in over 100 languages and it seems fair to claim that roughly 35% of the internet is powered by WordPress [1].

The expansive presence of WordPress makes the task of securing the CMS vitally important. Fortunately, WordPress is a relatively secure platform with a dedicated team of security professionals monitoring the core software for vulnerabilities and routinely releasing updates for addressing security issues [27]. Unfortunately, although a great many secure installations of WordPress exist, websites powered by this CMS still remain a popular target of attacks.

Attackers generally do not try to exploit flaws in the WordPress core. Instead, they make use of vulnerabilities that stem from two sources:

1) An average user of WordPress lacks the expertise to sufficiently secure their site, and often unknowingly leaves their website configuration in a vulnerable state.
2) WordPress, which was initially designed as a blogging engine, allows developers to extend the functionality of the CMS via third-party plugins.

The security of the third-party plugins is not guaranteed by WordPress itself, and indeed, 92% of the vulnerabilities found in WordPress powered websites are detected in third-party plugins [28]. WordPress users depend upon the useful features provided by these third-party plugins, therefore, plugin security demands primary importance.

In addition, the users' lack of expertise (in securing their WordPress websites) must also be accounted for when considering how to improve the overall security of WordPress powered websites. Compared to the users of other content management systems, WordPress users have less experience with security [25]. Advanced security tools (such as Nikto and WPScan) may be beyond the abilities, ambitions, or means of majority of the WordPress users.

To accommodate this lack of experience, a variety of security scanner tools are conveniently available as WordPress plugins. These security scanner plugins are meant to detect security vulnerabilities in other WordPress plugins. Thus, in the context of this particular work of ours, we are talking about two categories of WordPress plugins: (a) *feature-rich plugins*, which offer useful features, and (b) *security scanner plugins*, which are meant to detect and expose the vulnerable plugins.

This paper presents an analysis of the efficacy of affordable security tools that are easy to find, install, and utilize. As the third-party plugins are the primary source of vulnerabilities in WordPress powered websites [28], we particularly investigate the ability of the *security scanner plugins* in detecting vulnerable *feature-rich plugins*. The findings are derived from both quantitative and qualitative analyses of the performance of 11 independent security scanner plugins. These analyses are conducted on a testbed composed of a website (we created) having dependencies on 51 *feature-rich plugins* with known security vulnerabilities.

II. Background and Related Work

There are studies on the security and quality assessment of source code and other artifacts [15], [16], [19], [17], [18] of software systems in general using different security scanning tools. Recently, Ryan et al. [20] conducted a study to assess the efficacy of security scanners for Android applications. This study of ours is completely different from all these studies in its objective and procedure. Instead of assessing the software artifacts or source code, we assess the efficacy of the security scanner plugins in detecting vulnerabilities in WordPress-powered web application.

The core issues addressed by this paper are the vulnerabilities represented by the WordPress plugin ecosystem and the relative inexperience of its userbase. A study of existing literature elucidates the evolution of both these issues. When WordPress was first published in 2003, its co-author Matt Mullenweg stated that the core mission of WordPress was to "democratize publishing". This remains a central goal of WordPress today and is evident both in the popularity of WordPress and the relative inexperience of its userbase. The ability of WordPress to democratize publishing stems from the ease with which users can extend its functionality via community contributed plugins. These plugins allow what is essentially a blogging tool to power applications ranging from social community plugins to payment systems to medical patient portals [21].

The current WordPress marketplace offers over 52,000 plugins, a remarkable feat, but one that unfortunately comes at the cost of security [24]. While WordPress.org does have a rudimentary review process for plugins, a staggering number of vulnerabilities make it past this review [26]. Indeed, WordPress is the world's most attacked CMS [27] and plugins account for over 92% of detected vulnerabilities [28]. The existing literature clearly indicates that plugins are the most vulnerable aspect of WordPress. Therefore, vulnerable plugin detection must be included in the functionality of any viable WordPress security tool.

Why, given the security issues commonly associated with WordPress, does it remain so popular? Its popularity stems directly from the original intent of Matt Mullenweg- the democratization of publishing means that even inexperienced users can, through the WordPress plugin marketplace, build websites with an incredible range of complex functionality. One might imagine that giving users the power to easily create such complex sites without the need to hire relatively expensive professional developers, may result in both a userbase without significant technical experience as well as web applications whose creators are unaware of the security concerns associated with their apps. Indeed, surveys indicate that the WordPress userbase is both inexperienced and overconfident, a troubling combination. In a survey conducted by Norrie et al. in 2014, it is shown that 96% of WordPress users consider themselves to be "Developers" despite less than 14% having a degree in a programming related field [25]. In a survey sent to WordPress users identifying as "Developers" by the software development firm Delicious Brains, 91.6% of respondents are seen to come from disciplines unrelated to software development [23].

Given the large number of attacks on WordPress sites, one might surmise that there are not sufficient security tools available to users. In fact, there are many widely available tools that are easy to install and use. Additionally, several of the most popular tools claim to address the issue of plugin vulnerabilities. Unfortunately, existing literature largely focuses on the efficacy of more complex static analysis tools, but those tools are unlikely to be utilized by inexperienced users. There is a definite need for studies that analyze popular and easy to use tools, such as the Wordfence security plugin. By merit of their popularity, these tools have the most potential to reduce the number of WordPress exploits worldwide. The goal of this study is to conduct such an analysis. The methods for testing the efficacy of these popular plugins are outlined in the next section.

III. Methodology

A. Selection of Security Scanner Plugins

As the goal of this work is to examine the the efficacy of the security scanner plugins in exposing insecure/vulnerable plugins, in this study, we include only those security scanner plugins that match the following criteria.

- Plugin security – must claim to address plugin security
- Popular – must have 10,000+ active installs
- Easy to install – must be available in the WordPress.org plugin repository
- Affordable – must offer a free tier for users on a budget. (For the purposes of this study- only the free tier of a plugin is analyzed)

Based on the above criteria, we obtain 11 WordPress security scanner plugins for our study as presented in Table I. All these security scanner plugins are downloaded from https://wordpress.org.

TABLE I
Security Scanner Plugins Examined in this Study

Security Scanner Plugin	Version	# of Active Installs
Jetpack [8]	6.8	5,000,000+
WordFence Security [13]	7.4.12	3,000,000+
iThemes Security [7]	7.9.0	1,000,000+
All In One WP Security & Firewall [3]	4.4.4	900,000+
Bullet Proof Security [4]	4.3	60,000+
Security Ninja [2]	5.111	10,000+
Titan Anti-spam & Security [12]	7.2.1	100,000+
Sucuri Security [11]	1.8.24	700,000+
Defender Security [6]	2.4.5	40,000+
Cerber Security [5]	8.7	200,000+
SecuPress [10]	1.4.12	30,000+

Although this study is concerned with each security tool's ability to detect vulnerable plugins, it should be noted that each of these tools offer a variety of protections beyond plugin scanning, including reminders to update the WordPress core and plugins, regional IP block lists and more. These features do provide some level of security, but do not address

the primary threat represented by vulnerable plugins. Such features are therefore not considered further in this paper.

B. Selection of Vulnerable Plugins

The popular *Exploit Database* maintained by Offensive Security [9] is used to identify WordPress Plugins that had been flagged with known vulnerabilities over the past 2 years [9]. The majority were sourced from the wordpress.org directory, although some were downloaded from third-party websites such as github.com and directly from the Exploit Database.

Table II presents a list of the 51 feature-rich vulnerable/insecure plugins we choose along with their known security vulnerabilities. In the 51 vulnerable plugins we have selected, the most common vulnerability type is cross-site scripting at 37%, followed by SQL Injection at 20%. A full breakdown is shown in Figure 1.

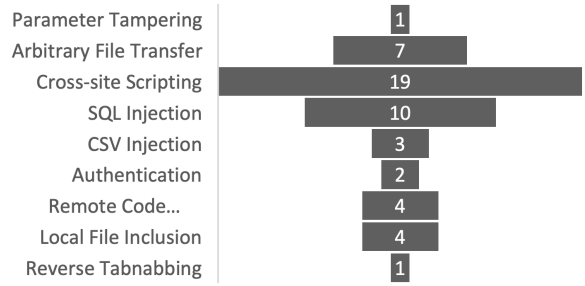

Fig. 1. Counts of WordPress plugins' most common vulnerability types

C. Testbed Setup

In this work, we use WordPress version 5.0. We set up a testbed based on the LAMP (Linux, Apache, MySQL, PHP) stack, which is commonly used to host WordPress sites. It is launched on an Oracle VM Virtual Box instance of Ubuntu 20.04.1. A WordPress powered website is hosted on Apache 2.4.41 running PHP 7.4.3 and MySQL 8.0.22. This website is configured to use the 11 security scanner plugins one at a time. The selected 51 WordPress plugins with known vulnerabilities are installed, then the VM state is saved utilizing VBox's snapshot function. This saved snapshot contains the state of the VM with fresh installation of those 51 vulnerable plugins. This is done to ensure that each security plugin being examined is installed on an identical instance of the WordPress site, which is a necessity as some security scanner plugins would permanently alter aspects of the WordPress installation, even after being uninstalled.

D. Assessment Procedure

Each security scanner plugin is evaluated separately. Before examining the ability of an individual security scanner plugin, the virtual machine's state is reset to the state with fresh installation of the vulnerable plugins. Then, the security scanner plugin is installed. Finally, the plugin's scan feature was run and its results are analyzed to determine which, if any, vulnerable plugins were detected.

IV. Findings

In this section, we present the assessment of the security scanner plugins and their comparative evaluation with respect to their efficacy in detecting the vulnerable plugins. It must be noted that the reported results from different security scanner plugins are not perfectly comparable. For example, two security scanner plugins, Security Ninja and WordFence, both flag an equal number of vulnerable plugins, but Security Ninja offers much more information regarding why a plugin was flagged. Thus, determining which ones perform "better" needs a subjective evaluation. We, therefore, first present the initial quantitive results in Section IV-A, and then we present a deeper qualitative assessment in Section IV-B.

A. Quantitative Results

In Figure 2, we present the number of vulnerable plugins detected by the 11 security scanner plugins examined in this study. Surprisingly, six out of those 11 security scanner plugins

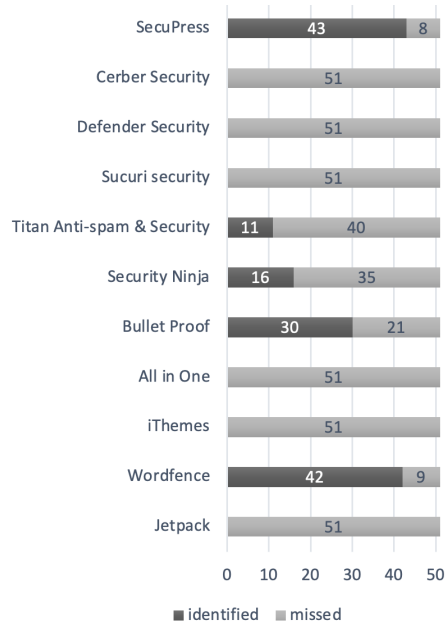

Fig. 2. Performance of Security Scanning Tools

completely failed in detecting any of the 51 vulnerable plugins. These six failing security scanner plugins are: Cerber Security, Defender Security, Sucuri Security, All in One WP Seurity, iThemes Security, and Jetpack.

All six of those security scanner plugins require users to upgrade to a premium, paid version of the plugin to access the capabilities that would scan the plugin directory. This phenomenon is further explored in Section IV-B.

For each of the remaining five security scanner plugins, we now describe how many vulnerable plugins are successfully detected and which security vulnerabilities existed in those vulnerable plugins that escaped detection.

SecuPress demonstrated the best performance, correctly identifying 43 vulnerable plugins, flagging 41 as "not up to date" and 2 as not "updated for at least 2 years". The scan

TABLE II
FEATURE-RICH WORDPRESS PLUGINS AND THEIR KNOWN VULNERABILITIES

Vulnerable Feature-rich Plugin	Version	Known Vulnerability
Ad Manager WD	1.0.11	Arbitrary File Download
Add Mime Types	2.2.1	Cross-Site Request Forgery
Adicon Server	1.2	'selectedPlace' SQL Injection
Advanced-Custom-Fields	5.7.7	Cross-Site Scripting
Ajax Load More	5.3.1	Authenticated SQL Injection
Anti-Malware Security and Brute-Force Firewall	4.18.63	Local File Inclusion (PoC)
Appointment Booking Calendar	1.3.34	CSV Injection
Audio Record	1.0	Arbitrary File Upload
AutoSuggest	0.24	'wpas_keys' SQL Injection
Autoptimize	2.7.6	Arbitrary File Upload
Baggage Freight Shipping Australia	0.1.0	Arbitrary File Upload
BBPress	2.5	Unauthenticated Privilege Escalation
Booking Calendar	8.4.3	(Authenticated) SQL Injection
Buddypress	6.2.0	Persistent Cross-Site Scripting
ChopSlider	3.4	'id' SQL Injection
Colorbox Lightbox	1.1.1	Persistent Cross-Site Scripting
Contact Form Builder	1.0.67	Cross-Site Request Forgery / Local File Inclusion
Contact Form Maker	1.13.1	Cross-Site Request Forgery
contact-form-7	5.1.6	Remote File Upload
Drag and Drop File Upload Contact Form	1.3.3.2	Remote Code Execution
Easy Testimonials	3.2	Cross-Site Scripting
FooGallery	1.8.12	Persistent Cross-Site Scripting
Form Maker	1.13.3	SQL Injection
Google Review Slider	6.1	'tid' SQL Injection
GoURL.io	1.4.14	File Upload
Helpful	2.4.11	SQL Injection
Import Export WordPress Users	1.3.1	CSV Injection
Insert or Embed Articulate Content into WordPress	1.0	Remote Code Execution
JoomSport	3.3	SQL Injection
Like Button	1.6.0	Authentication Bypass
Loco Translate	2.2.1	Local File Inclusion
Maintenance Mode by SeedProd	5.1.1	Persistent Cross-Site Scripting
Multi-Scheduler	1.0.0	Cross-Site Request Forgery (Delete User)
OneSignal	1.17.5	'subdomain' Persistent Cross-Site Scripting
PayPal Checkout Payment Gateway	1.6.8	Parameter Tampering
Plugin Media Library Assistant	2.81	Local File Inclusion
Popup Builder	3.49	Persistent Cross-Site Scripting
Postie	1.9.40	Persistent Cross-Site Scripting
Powie's WHOIS Domain Check	0.9.31	Persistent Cross-Site Scripting
Search Meter	2.13.2	CSV injection
Simple File List	4.2.2	Arbitrary File Upload Remote Code Execution
Simple Membership	3.8.4	Cross-Site Request Forgery
Sliced Invoices	3.8.2	'post' SQL Injection
Social Warfare	3.5.3	Remote Code Execution
ultimate-member	2.1.3	Local File Inclusion
Wisechat	2.6.3	Reverse Tabnabbing
WooCommerce Product Feed	2.2.18	Cross-Site Scripting
WOOF Products Filter for WooCommerce	1.2.3	Persistent Cross-Site Scripting
WP Google Maps	7.11.18	SQL Injection
WP Sitemap Page	1.6.2	Persistent Cross-Site Scripting
WPForms	1.5.8.2	Persistent Cross-Site Scripting
	1.6.3.1	Cross site scripting

failed to detect some plugins having the following vulnerabilities: SQL injection, CSV Injection, Cross-site scripting, and Parameter tampering.

Wordfence correctly identified 42 vulnerable plugins, flagging them as "threats". 37 plugins were listed as "critical threats" and four as "medium threats". The problem associated with every flagged plugin was of "type: Plugin Upgrade". The scan allowed some plugins with the following vulnerabilities to go undetected: SQL Injection, SCV Injection, Cross-site scripting.

Bullet Proof Security flagged 30 plugins as containing "suspicious files". It should be noted that Bullet Proof flagged a total of 1,342 files as "suspicious", but only 30 of them were located in plugins. The scan allowed some plugins with the following vulnerabilities to go undetected: SQL Injection, CSV Injection, Cross-site scripting, Privilege escalation, Remote code execution, Arbitrary File Download.

Security Ninja flagged 16 vulnerable plugins, listing the vulnerability type and linking to the relevant CVE page. The scan allowed some plugins with the following vulnerabilities

to go undetected: Parameter tampering, SQL Injection, CSV Injection, Cross-site scripting, Privilege escalation, Remote code execution, Arbitrary File Download.

Finally, Titan Anti-spam & Security successfully identified 11 vulnerable plugins, flagging suspicious files found in each plugin's directory. Additionally, Titan highlighted the suspicious code found in each file. It failed to identify some plugins with the following vulnerabilities: Arbitrary File Transfer, Cross-site Scripting, SQL Injection, CSV Injection, Local File Inclusion, Authentication, Parameter Tampering, Remote Code Execution, Reverse Tabnabbing.

B. Qualitative Assessment

The most alarming finding is that despite claiming to provide plugin security, six of the security tools studied (representing over 7,840,000 active installs) provide no capability for identifying vulnerable plugins under their free-tier. This is a popular strategy for "freemium" WordPress plugins, whereby the free-tier provides basic functionality but requires an upgrade to a paid-tier to access additional features [14].

The basic functionality offered by these plugins commonly includes features like block lists, login obfuscation, and monitoring the integrity of WordPress core files. These are all legitimate security features, but they fail to address the biggest threat to WordPress security, plugin exploits. Considering that installing any security plugin may offer a false sense of security, it seems reasonable to conclude that many users will not be motivated to upgrade to a paid-tier of these security tools, leaving millions of sites unknowingly unprotected against the most common threats.

Of the security tools that did successfully identify at least some vulnerable plugins, the results are challenging to quantify given that no two tools provide their results in the same format. Some of the best performing tools provide little to no information regarding the type or severity of the detected vulnerabilities. Other plugins had inferior detection rates, but did provide extensive threat descriptions. Since these descriptions provide a context that may inform a user's decision to upgrade or delete a vulnerable plugin, the utility of security tools cannot be based on their total identifications alone. Therefore, instead of a delineated ranking, a brief analysis addressing the merits of each tool is called for. We omit from consideration the tools that failed to provide free-tier plugin vulnerability support.

Titan Anti-spam & Security flagged 11 vulnerable plugins and provided details regarding the suspicious code that triggered the detection. Interestingly, in all 11 cases the suspicious code involved the use of the JavaScript eval function. While there are legitimate uses of the function, it is also true that eval (which allows JavaScript to execute arbitrary strings as code) is a common source of security vulnerabilities [22]. Unfortunately, the eval function was not the source of the reported vulnerability in the plugins identified. Additionally, it is unlikely that the average user is aware of the security implications of the eval function, limiting the utility of this information.

Although SecuPress performed the best in terms of number of detections, it provided limited context for the detections stating only that identified plugins were using an "out of date" version, or that no new versions had been released for over 2 years. Identifying plugins that haven't been updated for a long time is useful information, but simply stating that plugin is "out of date" (representing 41 of SecuPress' 43 detections) is less helpful. There are valid reasons for not upgrading a plugin to a newer version. For example, if the next version removes or breaks needed functionality while not offering any security updates, a developer could reasonably skip the update. Not being given a reason beyond "out of date" could lead a developer to make a false assumption that their version of the plugin is still secure. Further, this functionality (identifying out of date plugins) is already provided by the WordPress core.

Wordfence performed similarly, identifying the 41 plugins that were out of date as requiring a "plugin upgrade". As noted for SecuPress, this description provides limited utility.

Bullet Proof Security appears to perform well, identifying 30 vulnerable plugins. However, those plugin detections were a subset of 1,342 "issues" detected by the tool. These issues are all labelled as "Suspicious File" and the "Pattern Match" that caused the file to be flagged is listed. The task of identifying vulnerable plugins from such a large list is challenging, making Bullet Proof Security an inadequate tool for casual users.

Security Ninja correctly identified only 16 vulnerable plugins, but the details it provided are commendable. For example, the Media Library Assistant plugin was flagged with the following description: "The Media Library Assistant plugin before 2.82 for Wordpress suffers from a Local File Inclusion vulnerability in mla_gallery link=download."

This tool not only correctly identifies the source of the vulnerability, but also provides valuable context for the user, allowing them to choose an appropriate course of action.

V. Threats to Validity

Construct Validity and Internal Validity: In this work, we used a testbed composed of a dummy WordPress website having dependencies on the 51 vulnerable plugins. This setup may not ideally represent a typical usage scenario of those plugins. However, it enables us to examine the 11 security scanner plugins' abilities to detect those vulnerable ones. To prevent any side-effects of one security scanner plugin on another, before invoking each security scanner plugin, we reset the testbed and thus we made sure that each security scanner plugin is operated on an identical fresh testbed.

One may argue whether it could be better to have included paid versions of the commercial security scanner plugins. We deliberately chose not to go along that direction as affordable security has remained a prime criteria for this particular work of ours. We plan to conduct a separate study in future on the paid versions of the commercial security scanner plugins.

External Validity: Our study examines only WordPress security scanner plugins operated on 51 feature-rich vulnerable WordPress plugins. Thus the results of this work may not be

generalizable to the variety of content management systems other than WordPress. Recall that, this work particularly focused on WordPress powered websites, since WordPress is the most popular content management system, which claims 60.3% of all content management systems in use [24].

Our testbed includes only 51 WordPress feature-rich plugins with known vulnerabilities on which 11 security scanner plugins are separately operated one by one. One may argue that the results based a dataset of this size may not be generalizable enough. While it is true that a larger dataset could yield higher confidence in generalizability, our dataset, having been drawn from a popular *Exploit Database* [9], helps in maintaining high reliability/reproducibility of this work. Moreover, this dataset includes the vulnerable plugins recently flagged over past two years [9].

Reliability: The methodology of data collection, analysis, and results are well documented in this paper. All the security scanner plugins and the vulnerable feature-rich plugins are publicly available online. The particular versions of these plugins used in this work are also clearly mentioned in Table I and in Table II. Hence, it should be possible to replicate this study.

VI. Conclusion

In this paper, we have presented an empirical assessment of the WordPress security scanner plugins' efficacy in the detection of vulnerable feature-rich plugins. We have examined 11 WordPress security scanner plugins by operating them separately on a testbed composed of a WordPress powered website with dependency on 51 feature-rich plugins having known security vulnerabilities.

We have found that performances of the security scanner plugins largely vary in their sensitivity to insecurities in the vulnerable plugins. We have also found that some security scanner plugins are more expressive than others in their reporting of the detected vulnerable plugins.

While the WordPress core itself is known to be relatively secure, the vast WordPress plugin eco-system is certainly not, indicating a significant threat to around 455 million WordPress powered websites [1] and their billions of viewers. Unfortunately, all the security scanner plugins analyzed in this study failed to adequately address this threat.

The most striking finding of this study is that, at the time of writing, there is not a single free WordPress security scanner plugin capable of sufficiently detecting and flagging plugin vulnerabilities. At least, we were not able to find such a free-tier WordPress plugin. This is surprising considering that each vulnerability targeted in this study is listed on the publicly viewable site at exploit-db.com [9]. Given this result, one may reasonably surmise a connection between WordPress being possibly the most attacked CMS in the world, and the failure of the most popular security tools in alerting WordPress users to even publicly known plugin vulnerabilities.

References

[1] *2020's Most Surprising WordPress Statistics*. https://www.whoishostingthis.com/compare/wordpress/stats/, 2021.

[2] *Security Ninja*. https://wordpress.org/plugins/security-ninja/, 2021.

[3] *All In One WP Security & Firewall*. https://wordpress.org/plugins/all-in-one-wp-security-and-firewall/, verified: May 2021.

[4] *Bullet Proof Security*. https://wordpress.org/plugins/bulletproof-security/, verified: May 2021.

[5] *Cerber Security*. https://wpcerber.com, verified: May 2021.

[6] *Defender Security*. https://wordpress.org/plugins/defender-security/, verified: May 2021.

[7] *iThemes*. https://ithemes.com/security/, verified: May 2021.

[8] *Jetpack*. https://jetpack.com, verified: May 2021.

[9] *Offensive Security Exploit Database Archive*. http://exploit-db.com, verified: May 2021.

[10] *SecuPress*. https://secupress.me, verified: May 2021.

[11] *Sucuri Security*. https://sucuri.net, verified: May 2021.

[12] *Titan Anti-spam & Security*. https://wordpress.org/plugins/anti-spam/, verified: May 2021.

[13] *Wordfence*. https://www.wordfence.com, verified: May 2021.

[14] Jordi Cabot. Wordpress: A content management system to democratize publishing. *IEEE Software*, 35(3):89–92, 2018.

[15] M. Islam and M. Zibran. A comparative study on vulnerabilities in categories of clones and non-cloned code. In *Proceedings of the 10th IEEE International Workshop on Software Clones*, pages 8–14, 2016.

[16] M. Islam and M. Zibran. On the characteristics of buggy code clones: A code quality perspective. In *Proceedings of the 12th IEEE International Workshop on Software Clones*, pages 23 – 29, 2018.

[17] M. Islam and M. Zibran. How bugs are fixed: Exposing bug-fix patterns with edits and nesting levels. In *Proceedings of the 35th ACM/SIGAPP Symposium on Applied Computing*, pages 1523–1531, 2020.

[18] M. Islam and M. Zibran. What changes in where? an empirical study of bug-fixing change patterns. *ACM Applied Computing Review*, 20(4):18–34, 2021.

[19] M. Islam, M. Zibran, and A. Nagpal. Security vulnerabilities in categories of clones and non-cloned code: An empirical study. In *Proceedings of the 11th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, pages 20–29, 2017.

[20] R. Joseph, M. Zibran, and F. Eishita. Choosing the weapon: A comparative study of security analyzers for android applications. In *Proceedings of the International Conference on Software Engineering, Management and Applications*, pages 1–7 (to appear), 2021.

[21] T. Koskinen, P. Ihantola, and V. Karavirta. Quality of wordpress plug-ins: An overview of security and user ratings. In *Proceedings of International Conference on Privacy, Security, Risk and Trust (PASSAT)*, pages 834–837. IEEE Computer Society, 2012.

[22] Fadi Meawad, Gregor Richards, Floreál Morandat, and Jan Vitek. Eval begone!: Semi-automated removal of eval from javascript programs. In *Proceedings of International Conference on Object-Oriented Programming, Systems, Languages & Applications*, pages 607–620, 2012.

[23] WP Offload Media. *How Much Money Do WordPress Developers Make? That and More Insights on the Life of a WordPress Developer in Our First-Ever Industry Report*. https://deliciousbrains.com/wordpress-developer-statistics/, 2019.

[24] Oslien Mesa, Reginaldo Vieira, Marx Viana, Vinicius H. S. Durelli, Elder Cirilo, Marcos Kalinowski, and Carlos Lucena. Understanding vulnerabilities in plugin-based web systems: An exploratory study of wordpress. In *Proceedings of the 22nd International Systems and Software Product Line Conference-Vol. 1*, pages 149–159, 2018.

[25] Moira C. Norrie, Linda Di Geronimo, Alfonso Murolo, and Michael Nebeling. The forgotten many? a survey of modern web development practices. In *Proceedings of the 14th International Conference on Web Engineering, Lecture Notes in Computer Science, vol 8541*, pages 290–307. Springer International Publishing, 2014.

[26] Jukka Ruohonen. A demand-side viewpoint to software vulnerabilities in wordpress plugins. In *Proceedings of the Evaluation and Assessment on Software Engineering (EASE)*, pages 222–228. ACM, 2019.

[27] Hannes Trunde and Edgar Weippl. Wordpress security: An analysis based on publicly available exploits. In *Proceedings of the 17th International Conference on Information Integration and Web-Based Applications & Services*, pages 1–7. ACM, 2015.

[28] James Walden, Maureen Doyle, Grant A. Welch, and Michael Whelan. Security of open source web applications. In *Proceedings of the Third International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pages 545–553. IEEE Computer Society, 2009.