Hidden in Plain Sight: An Empirical Study of Malware in Nulled WordPress Plugins

Shubham Pharande
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta Georgia USA
spharande3@gatech.edu

Atharav Hedage
School of Cybersecurity and Privacy
Georgia Institute of Technology
Atlanta Georgia USA
ahedage3@gatech.edu

Sanyam Pandey
School of Cybersecurity and Privacy
Georgia Institute of Technology
Atlanta Georgia USA
spandey90@gatech.edu

Abstract

Content management system (CMS) are software applications that allow users to create, manage, and publish digital content, typically for websites, WordPress being the most popular among all. Many CMSes offer plugins and themes that extend the functionality and appearance of the CMS. Modern websites owe a lot of their aesthetics and functionalities to the simplicity and affordances of Content Management Systems (CMS), and the plugin ecosystem that comes with it. It has been estimated that over 55% of the websites online today are powered by various CMS platforms. Unfortunately, this large, technically-novice user base has made the CMS-based websites and web servers, an important target for hackers, and CMS plugins, being user-controlled installations, have gained importance as an easy way to gain access to the website and web server. These plugins are actively bought and sold on widely popular marketplaces. Driven by these economic incentives that come along with malicious access, attackers abuse this economy – by pirating popular plugins, ingesting them with malware, and either selling them for a lower price, or giving them away for free using various distribution channels.

This research studies three such distribution channels – Web-based marketplaces, Telegram channels, and Torrents. We studied over 4000 plugins sampled across various marketplaces on the aforementioned three distribution channels, and identified *1851* malicious plugins, showing a wide variety of threat levels for the owners and users of the website. We developed Kratos, an automated framework to scan plugins downloaded from potentially malicious origins and provide a friendly report on malicious behaviors detected in the plugin code.

Introduction

Content Management System (CMS) platforms power over 68% of the world's websites, with WordPress alone controlling nearly 60% of the CMS market, with an overall share of 43.2% [6]. This widespread adoption has led to a rapid increase in cyberattacks targeting CMS-based websites. CMS ecosystems also allow for other affordances to help make website-building simpler – modular and reusable code for adding and improving functionality (plugins) and design and user experience (themes), which are widely adopted by CMS-based websites. Plugins are add-ons that provide additional features or functionality, such as SEO optimization or e-commerce integration, while themes are pre-designed templates that can be applied to change the visual appearance of a website. Using these plugins and themes places an implicit trust in unvetted code with limitless access to the webserver or the ecosystem. This trust is abused by malware authors as an easy way to get access to the website and the server if possible. At the same time, since customers are also ready to pay for access to good plugins, malware authors often sell malicious plugins

for monetary gains as well. The malicious code in these plugins do not employ evasion techniques or obfuscation, brazenly *hiding in plain sight*.

Despite their significant impact, little research has been conducted in CMS-targeting malware to-date. The traditional way of studying malware has been via instrumentation and fine-grained logging to understand its *modus operandi*. However, this is difficult in the CMS domain, specifically in the case of studying modular plugins, given the complexity of setup required to replicate their behavior, the performance overhead notwithstanding. The overarching goal of a CMS is to simplify the process of creating, managing, and publishing digital content. Thus often, the website owners might not be technically adept to adopt such monitoring solutions, and they might not even have access to the underlying webserver in the first place, which might be owned and maintained by a hosting provider. Anti-virus scanners are deployed to scan websites for malware, but this approach has rarely worked in tackling web malware in practice, with a high false-positive rate leading to real alarms being ignored [1,2].

Nulled plugins refer to pirated versions of originally paid plugins that are freely distributed through nulled marketplaces without the permission of the original creator [3]. Nulled plugins are often created by hacking or modifying the code of original paid plugins in order to bypass the license key verification and enable indefinite use of the plugin without payment. However, these modifications often introduce vulnerabilities or malicious code into the plugin, which can harm users or collect sensitive data without their knowledge or consent. Note that not all nulled plugins portray malicious behaviors.

The average price of the most commonly used nulled plugins is around $59, with a price range of $36 to $89 per plugin [3]. Legitimate plugin marketplaces such as the official WordPress Plugin Directory, CodeCanyon, ThemeForest and WooCommerce provide a secure and trustworthy platform for users to browse, purchase, and download plugins. Here, the plugins range from $2 to $63 for best-selling ones. These marketplaces are considered safe because they have strict guidelines and policies in place to ensure the safety and quality of the plugins they offer. For instance, the WordPress Plugin Directory provides a repository of free plugins that are regularly reviewed and tested for compliance with WordPress coding standards and best practices. Additionally, any plugin that violates guidelines or poses a security threat is promptly removed. However, the ease of access and the potentially lucrative economic incentives have led to a proliferation of non-legitimate marketplaces, many of which distribute pirated or malware-infected plugins. These shady marketplaces can be found on the web, through social media channels like Telegram, and via torrent sites, and are often promoted through simple search queries. We found that these channels are easily accessible through simple Google and Telegram queries, it was also found that there are various YouTube tutoring videos, blogs and forums where these websites are promoted. A simple search of “WordPress Plugin” on torrent search engine (1377x[.]to) gives 50 pages as result consisting of 20 plugins in each page till date. Overall, Nulled Plugin distribution websites were the most easily accessible.

Our research involved downloading these nulled plugins from these three distribution sources and then analyzing these plugins to identify the percentage, severity and likelihood of the risk involved with each distribution source. Combined these three datasets, we downloaded 60 gigabytes of data. Our analysis involved using regex based and AST based signatures to traverse the plugins code and flag them based on their behavior. The signatures were built through manual analysis of plugins.

Our study analyzed 4271 plugins in total of which we identified 1851 plugin as malicious. It was found that the distribution websites accounted for the highest number of plugins accessible. The probability that a user is downloading a malicious plugin in the form of nulled plugin is approximately 46%.

Related Work

"Mistrust Plugins You Must: A Large-Scale Study of Malicious Plugins In WordPress Marketplaces" [3] focuses on identifying and analyzing malicious plugins in WordPress marketplaces, emphasizing the importance of careful plugin selection and vetting. However, this research does not talk about distribution of malware over various sources. "Understanding vulnerabilities in plugin-based web systems: An exploratory study of WordPress" [7] investigates the different types of vulnerabilities in WordPress plugins, including XSS and SQL injection. This research gave valuable insights about analyzing specific vulnerabilities in plugin-based systems.

"A demand-side viewpoint to software vulnerabilities in WordPress plugins" [5] examines the demand-side of vulnerabilities, considering factors such as user adoption and plugin popularity. "Dictionary attack on WordPress: Security and forensic analysis" [4] discusses the use of security plugins as a means of enhancing the security of WordPress websites. These plugins can help to identify and block suspicious login attempts, as well as provide additional security features such as malware scanning and firewall protection. Overall, these papers highlight the importance of plugin security and the need for developers and users to be vigilant in preventing and addressing vulnerabilities.

Methodology

Preliminary Investigation

To understand the prevalence of the problem, we first manually collected data from all three sources. Since manual investigation would involve a lot of searching and time, we focused on downloading plugins that had more chance of being malware – those associated with e-commerce and user data. We downloaded 89 plugins in total and did a round of manual analysis on them. This manual analysis included reading-through and understanding the PHP code involved with loading the module and its custom components and running a VirusTotal scan on the plugin. As expected, VirusTotal did not flag malicious behaviors in the code, although in some cases, two or less engines (out of 70) flagged ad injection malware. Our manual investigation of the code helped us identify the malicious behaviors as well as code patterns which we used to build the malware signatures for Kratos.

Data Collection

The primary objective of our paper is to identify major distribution mechanisms, discover their ease of use, and understand the prevalence of malware in nulled plugins. Additionally, we intend to classify the types of malwares and understand their impact on webmasters and end-users.

To achieve our objectives, we have used three different datasets: Torrent, Nulled Marketplaces, and Telegram. We chose Torrent as the dataset because it often distributes illegal copies of software, themes, plugins, and other digital products used to build and maintain websites and servers. To avoid the repetition of plugins and themes, we selected three major torrent search engines which accounted for the majority of data: www[.]1337x[.]to, katcr[.]to, and thepiratebay[.]party. We created crawlers to auto-download the plugins and themes, but we faced some limitations due to less availability of seeders, which made it difficult to download the entire dataset. Additionally, we also faced barriers of rate limit which we bypassed using random delays. Our research project involved the manual analysis of plugins downloaded from torrent. During the analysis, we observed that the plugin authors had included links to their distribution channels in the code. For example, we found remarks like "Visit us on www[.]nulled[.]ch" in the codes and readme files. By tracing these remarks, we were able to identify several new distribution websites and sources that

were being used for distributing malware-infected plugins. This trace made us aware of the existence of one of the major distribution sources, namely, Telegram. Overall, the tracing helped us in broadening our data set from Torrent to Websites to Telegram.

We expanded our scope to downloading plugins and themes from nulled marketplaces. We selected three websites: www[.]vestathemes[.]com, www[.]wptry[.]org and www[.]wplocker[.]com and we created individual crawlers to download plugins and themes from these websites. During the course of our research, we encountered various challenges in building crawlers for distribution websites. One of the main challenges was the lack of standardization in website structures, which made it difficult for our crawlers to traverse the websites continuously. As a result, we had to restructure our crawlers for each unexpected error encountered while crawling the websites. Despite these challenges, we were able to develop a robust crawling system that enabled us to collect a large number of plugins from the distribution websites for analysis. Further we encountered some websites that have statistically embedded download links, while others had a series of fetch requests to obtain the final download link. The series of fetch requests were attempted to be kept hidden, but our manual analysis was able to uncover this tactic.

By tracing the remarks left by authors in the dataset collected using Torrent and Websites, we selected Telegram channels as our next source. This led us to check Telegram, where we found and downloaded more than 800 plugins and themes.

Malicious Behavior Detection

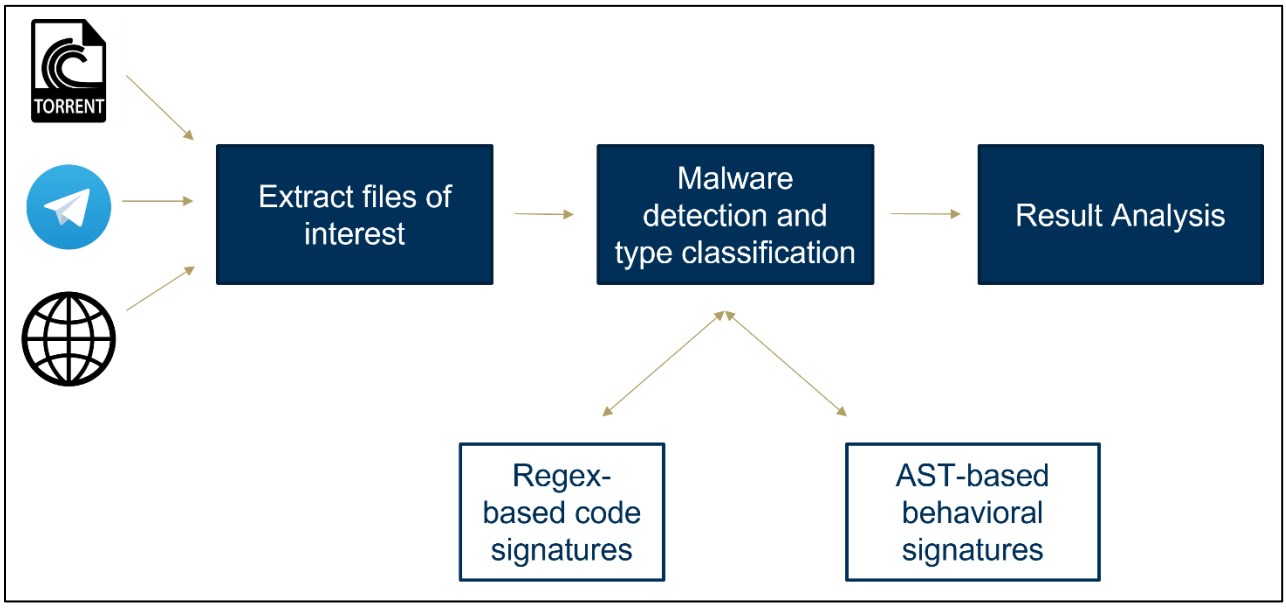

Figure 1: Schematic of the Kratos pipeline

Figure 1 shows a schematic representation of our research methodology. Our pipeline consists of three major components – extraction of files of interest, malware detection and classification and result compilation.

Extraction of files of interest

Plugins contain a wide variety of files, including code, images, configuration files and rarely binaries. Most of these files do not contain malware, especially relevant to our problem statement. Thus, to avoid scanning every file, Kratos targets PHP malware alone.

To clean and structure the dataset obtained for our research, we encountered several challenges, such as unnecessary files and varying zip extensions, along with the base directory being deeply nested within subfolders. To address these issues, we developed a Python script that effectively extracted the nested files and subsequently isolated the base directory. This streamlined the process of cleaning the dataset.

Further, to identify the files of interest accurately, we make use of the magic module in Unix [8]. Magic uses the first four bytes of the file to identify the file type and MIME type. Thus, we do not rely on file extensions to identify file types, which can be trivially modified by malware authors without any impact on functionality.

Additionally, Kratos also uses regex-based string matching to identify plugin metadata from code files. WordPress requires all plugins to have a special header (in the form of a block comment) to enable loading the plugin into the website. If an attacker tries to evade our detection by dropping the header, the plugin won't be loaded into the website. This header includes metadata like plugin name, version, plugin URI, author name and author URI.

Malware detection and classification

To identify the presence of malware in PHP code, we rely on signatures that were derived from our manual investigation. We use two types of code signatures – syntactic features (like regex-based search for code patterns, use of sensitive APIs, etc.) and semantic features (Abstract Syntax Tree – AST – based behavioral analysis). Regex-based search looks for specific strings in code that we identified based on our manual analysis. These strings include custom function names, specific strings used by specific authors, domain patterns, etc. AST-based analysis, on the other hand, identifies specific behaviors shown by the code. For example, the typical flow for a code block that downloads binaries from a remote server is as follows – using one of the PHP HTTP GET APIs (file_get_contents, wp_remote_get, http_get) or start a curl session in PHP, with the first argument of the function call being a string (URL – either as a variable or directly specified). This URL is then extracted and verified for malware via VirusTotal. AST helps codify this behavior. The full list of signatures used in Kratos is as follows:

1. **PHP and WP API Abuse** – Detects the use of WP and PHP functions used for potentially malicious purposes, including the following – disabling plugins, admin user enumeration, malicious post insertion, spam downloader, admin user creation, user information access, etc.
2. **Blackhat SEO** – URL Injection in case visitor user agent is a search engine crawler
3. **Downloader** – Download using either GET requests, curl sessions or PUT requests from a known malicious URL (flagged by VirusTotal)
4. **Function Construction** – Using PHP's create_func to create a function that internally performs an eval()
5. **Input Gating** – Passwords are hardcoded as super global variables, and if the variable is set, certain code is conditionally executed. Benign plugins use credentials from website database for conditional code execution
6. **MPlugin** – A malicious module that injects ads for non-admin site visitors, and hides itself from the plugins list. Regex-based rule

7. **Spam Injection** – Downloading content from a known malicious domain and injecting content into the HTML output

Lastly, Kratos was bundled into a Docker image, so that it could be run as a container. Containerization helped us in a few ways – it created a fixed baseline environment to run all the analyses, one which could be destroyed and recreated to isolate the analysis of each plugin, thus removing the possibility of any malicious impact of any binaries bundled with the plugin, ensuring the reproducibility of each analysis run. It also helped us create a sandbox environment isolated from the machine on which these analyses were run.

Findings

Our research focused on understanding the answers to three research questions –

1. How prevalent is malware in nulled WordPress plugins in the three distribution mechanisms?
2. How is malware in the nulled WordPress plugin ecosystem different across the three distribution mechanisms?
3. Can this malware be classified into categories?

Towards this, we present the following findings:

Out of the 4271 plugins analyzed, we found over 1851 plugins to be malicious. Web had the highest share of these, with 1673 plugins being flagged as malicious, followed by 12 from Torrents and 166 from Telegram. Thus, over 52% of the plugins downloaded from the three web marketplaces were flagged by our tool as malicious. Telegram stood at 18.5% and Torrents at 13%. The last result was surprising, given that Torrents have a reputation for peddling malware. In many instances, the same plugin was seen to house multiple types of malware, which points towards the work of multiple actors, or an ecosystem of malware authors and distributors.

Diving deeper into the results from the web marketplaces, we can see that WpTry has the largest malicious plugin count at 1070 (77.5% of the total downloads). 1067 out of 1070 have hits on the MPlugin rule, which aims at ad injection, although, the domains used by the malware in these plugins are different from the one we saw in our ground truth. This points to the possibility that the author of MPlugin has a stake in the way this marketplace is run. The other major rule hits were on downloaders from known malicious domains, and administrator user enumeration. The next largest marketplace was Vestathemes, with 593 malicious plugins out of 1307 (45.3%). The largest rule hit is again MPlugin, infecting 591 out of 593 plugins. The next largest malware seen were admin user enumeration, SEO injection, malicious downloads and password-protected code execution. Lastly, WP Locker showed 10 out of 27 plugins to be malicious, all focusing on downloading malware.

On the other hand, plugins from Telegram channels had zero MPlugin hits, most focusing on malware downloads from external sites (103/166 – 62%), the others being admin user enumeration and SEO injection. Similarly, Torrents focused on malicious downloads to be the largest (6/12 – 50%), followed by admin user enumeration and spam content injection.

Platform	Source	Downloaded Plugins	Malicious Plugins	Largest type
Telegram	-	887	166	Downloader
Torrents	-	91	12	Downloader
Web	**Total**	**3293**	**1673**	**MPlugin**
	www[.]vestathemes[.]com	1307	593	MPlugin
	www[.]wplocker[.]com	27	10	Downloader
	www[.]wptry[.]org	1959	1073	MPlugin

Table 1: Overview of types of malwares across different distribution mechanisms

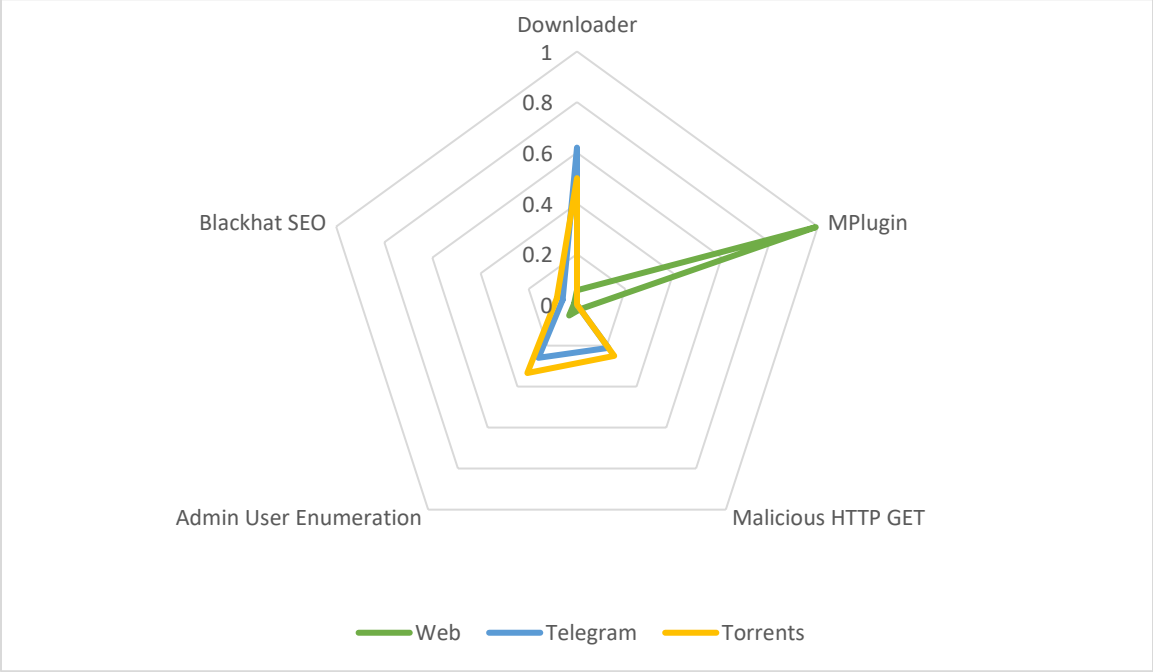

Figure 2: Malware types and difference across Web, Torrents, and Telegram (values in percentages)

Figure 2 shows the differences between malware seen across the three distribution channels. The web shows a large number of plugins with ad injection (MPlugin), while it's count is zero for the other two channels. On the other hand, plugins on Telegram and Torrent show a large prevalence of code that either downloads malware or makes GET requests to known malicious domains, followed by admin user enumeration. This shows a stark difference in motivations of malware authors who use these different channels. This also points towards the difference in risk to the website owners and users from these malicious plugins, from relatively benign on the web to more serious security risks from other sources.

Discussion

With WordPress being the backbone of a large percentage of websites today, given our observations, we can see that a large majority of websites online today are potentially compromised. Add-ons like WooCommerce and Shopify make it very easy to convert your WordPress website into a storefront. Our results are even more alarming in this context of e-commerce, where the malware can have access to your payment information and can act as a digital skimmer.

As a user, there is not a lot that you can do. But as a website owner, a few steps in hygiene can help. The first one being that plugins and themes should only be downloaded from marketplaces which vet the code for malware. Using nulled plugins from underground marketplaces can lead to large financial and reputation losses. Next, keep monitoring the website and web server for unauthorized changes. Using git-enabled daily backups can help the website owner monitor changes in a granular manner, and web server logs can help establish provenance. Some of these methods cannot be applied in case of the website being taken care of by a third-party hosting provider.

A basic search using the term "WordPress plugin" on any of these platforms yields a plethora of plugins that can be downloaded with relative ease, even by non-technical users. This easy availability and the volume of nulled plugins signifies the likelihood of a common user trying to download a malicious plugin.

There is a lot of potential for future work in this space, given that it has largely been ignored by the web security community. We have established that there is potential for data exfiltration using malicious plugins. Follow-up research could include identifying ways in which malware can and does steal data, ways by which they exfiltrate information out of the server, identifying command-and-control (C2) servers and trying to identify major actors in this space. Another direction of work spawning from this could include forensic research on understanding the use of compromised servers in relatively benign activities like phishing or redirect chains, or more malicious activities like botnet nodes or as C2 or malware staging servers.

In addition to our analysis, it is important to note that our dataset is limited in scope. Due to time and resource constraints, we restricted ourselves and were only able to analyze a portion of the available plugins. As a result, there is a large amount of downloadable data that remains unanalyzed. In the interest of further research and collaboration, we have made our crawler tools open source, which allows anyone to download and analyze these plugins. This provides an opportunity for future researchers to expand upon our findings and contribute to the understanding of the distribution of malware in plugin markets.

Conclusion

In conclusion, this research aims to provide a comprehensive study on malware distribution in WordPress plugins via Nulled Marketplaces, Telegram and Torrents. The study used a combination of datasets, tools, and techniques to analyze the prevalence, types, and impacts of malware. Through the analysis, it was found that a large percentage of WordPress websites potentially hold the risk of being compromised due to the use of nulled plugins and themes, and the malware can have access to payment information and act as a digital skimmer in the context of e-commerce.

Despite some limitations in the methodology, the project contributes to raising awareness among webmasters and end-users regarding the risks associated with using Nulled plugins and themes and providing statistical findings. As website owners, implementing good hygiene practices such as avoiding the use of Nulled plugins and themes, keeping plugins and themes up-to-date can help reduce the risk of malware infection for both the users and website owners.

Limitations

There are some limitations to our methodology. Firstly, we relied on publicly available resources, which might not be representative of all distribution mechanisms. Additionally, the use of anti-virus software, static code analysis, and dynamic analysis tools might not be sufficient to identify all types of malwares. Finally, we analyzed the plugins and themes on a standalone basis and did not consider their interaction with other plugins or themes, which might affect their behavior.

Ethical Considerations

During the study, we followed ethical guidelines and ensured that we did not engage in any illegal activities. We did not use any credentials or authentication to download the plugins and themes. Instead, we used publicly available resources such as Torrent, Nulled Marketplaces, and Telegram channels. Additionally, we have used the downloaded plugins and themes only for research purposes which are confined in a VM and did not distribute or share them for personal use or benefit.

References

[1] Advanced Threat Analytics. New Research from Advanced Threat Analytics Finds MSSP Incident Responders Overwhelmed by False-Positive Security Alerts. Retrieved April 23, 2023 from https://www.prnewswire.com/news-releases/new-research-from-advanced-threat-analytics-finds-mssp-incident-responders-overwhelmed-by-false-positive-security-alerts-300596828.html

[2] Wajih Ul Hassan, Shengjian Guo, Ding Li, Zhengzhang Chen, Kangkook Jee, Zhichun Li, and Adam Bates. 2019. NoDoze: Combatting Threat Alert Fatigue with Automated Provenance Triage. *Network and Distributed Systems Security Symposium* (February 2019). Retrieved April 23, 2023 from https://par.nsf.gov/biblio/10085663-nodoze-combatting-threat-alert-fatigue-automated-provenance-triage

[3] Ranjita Pai Kasturi, Jonathan Fuller, Yiting Sun, Omar Chabklo, Andres Rodriguez, Jeman Park, and Brendan Saltaformaggio. Mistrust Plugins You Must: A Large-Scale Study Of Malicious Plugins In WordPress Marketplaces.

[4] Ar Kar Kyaw, Franco Sioquim, and Justin Joseph. 2015. Dictionary attack on Wordpress: Security and forensic analysis. In *2015 Second International Conference on Information Security and Cyber Forensics (InfoSec)*, 158–164. DOI:https://doi.org/10.1109/InfoSec.2015.7435522

[5] Jukka Ruohonen. 2019. A Demand-Side Viewpoint to Software Vulnerabilities in WordPress Plugins. In *Proceedings of the Evaluation and Assessment on Software Engineering* (EASE '19), Association for Computing Machinery, New York, NY, USA, 222–228. DOI:https://doi.org/10.1145/3319008.3319029

[6] Usage Statistics and Market Share of Content Management Systems, April 2023. Retrieved April 23, 2023 from https://w3techs.com/technologies/overview/content_management

[7] Understanding vulnerabilities in plugin-based web systems | Proceedings of the 22nd International Systems and Software Product Line Conference - Volume 1. Retrieved April 28, 2023 from https://dl.acm.org/doi/abs/10.1145/3233027.3233042

[8] magic(5): file command's magic pattern file - Linux man page. Retrieved April 26, 2023 from https://linux.die.net/man/5/magic