

SCANTRAP: Protecting Content Management Systems from Vulnerability Scanners with Cyber Deception and Obfuscation

Daniel Reti¹^a, Karina Elzer¹^b and Hans Dieter Schotten^{1,2}^c

¹German Research Center for Artificial Intelligence, Kaiserslautern, Germany

²Technische Universität Kaiserslautern, Germany

Keywords: CMS, Penetration Testing, Vulnerability Scanner, Information Security, Cyber Deception, WordPress, Plugin, Website.

Abstract: Every attack begins with gathering information about the target. The entry point for network breaches are often vulnerabilities in internet facing websites, which often rely on an off-the-shelf Content Management System (CMS). Bot networks and human attackers alike rely on automated scanners to gather information about the CMS software installed and potential vulnerabilities. To increase the security of websites using a CMS, it is desirable to make the use of CMS scanners less reliable. The aim of this work is to extend the current knowledge about cyber deception in regard to CMS. To demonstrate this, a WordPress Plugin called 'SCANTRAP' was created, which uses simulation and dissimulation in regards to plugins, themes, versions, and users. We found that the resulting plugin is capable of obfuscating real information and to a certain extent inject false information to the output of one of the most popular WordPress scanners, WPScan, without limiting the legitimate functionality of the WordPress installation.

1 INTRODUCTION

Content Management Systems (CMS) have become a notable part of today's web, with almost half of the websites using some sort of CMS. This is due to the ease of use and free option. Specifically, out of the existing web CMSs, some popular CMSs have emerged: WordPress (WP) is the most popular CMS, with a market share of 63.3%. Shopify, Wix and Squarespace, as paid alternatives, together have 12.4% of the market share. The most popular open-source alternatives, Joomla and Drupal, are used in 4.3% of websites using CMS. (W3Techs, 2022)

With such popularity, however, they also become an attractive target for attackers. Due to the addition of third-party extensions, these platforms leave a large attack surface. In the field of penetration testing, different tools have emerged that reduce the effort of scanning and exploitation of popular CMSs. This makes reconnaissance possible with low effort, enabling the threat from so-called 'script kiddies' and bots.

Therefore, it is desirable to increase the effort of attackers in regards to attacking CMS by using different defence strategies. This starts with the attacker's capability of scanning CMSs. The approach of this work is based on cyber deception to disguise valuable information, by hiding the real and presenting false information.

As the most popular CMS as of today, we choose WP as a representative CMS to apply deception measures for our research. Due to its popularity, almost 100,000 plugins and 25,000 themes exist for WP. This leaves a large attack surface leading to over 30,000 known vulnerabilities. 92% of which are found in plugins. (WPScan, 2022) Also, because of its popularity, we choose WPScan, a WP vulnerability scanner maintained by Automattic, the vendor of wordpress.com, as the attacker's tool of choice.

This paper presents a systematic approach to deception based defence against CMS scanners. Hereby a WP plugin to defend against WPScan is created as a proof of concept. This plugin, called 'SCANTRAP', uses simulation and dissimulation techniques of the cyber deception domain to make reconnaissance harder for attackers. The source code can be found at <https://github.com/dfki-in-sec/SCANTRAP>.

Additionally, we give an overview of cyber decep-

^a <https://orcid.org/0000-0001-8071-6188>

^b <https://orcid.org/0000-0002-3984-779X>

^c <https://orcid.org/0000-0001-5005-3635>

tion methods for the web domain, the functionality of different CMS scanners and a survey of cyber deception methods used by existing WP security plugins.

Our paper is divided into six sections. The following section gives a brief overview of the background of cyber deception and CMS. The third section describes our created plugin. In the fourth, we present our discussion and in section five we examine related work. Our conclusion is drawn in the final section.

2 CONTENT MANAGEMENT SYSTEM SECURITY AND CYBER DECEPTION

In the cyber kill chain, a model describing the stages of an attack, the first phase is reconnaissance. In this phase, the attacker will gather information that will allow clearer decision-making in later steps. One method of reconnaissance is scanning. This paper's focus lies on defending this part of the cyber kill chain through cyber deception in the domain of CMS, which will be explained in detail in this section.

2.1 Cyber Deception

In the field of cyber security, the term *cyber deception* is generally understood as a defence strategy which is used to intentionally mislead an attacker. Specifically, by simulating fake or hiding existing targets, the actions and in-actions of the attacker should become more predictable or take more time. (Yuill et al., 2007)

In the literature, many deception techniques in multiple computer domains have been proposed and studied (Fraunholz et al., 2018a).

In 1991, Bell and Whaley defined a taxonomy for deception that contains two main classes (Bell and Whaley, 1991). Dissimulation describes the hiding of the existing and simulation refers to faking.

The term 'obfuscation' is a type of cyber deception and dissimulation, mainly understood in relation to code protection. However, the idea is to transform a structure to make it harder for an attacker to determine the functionality. (Behera and Bhaskari, 2015)

More concrete examples of cyber deception are honeypots and honeytokens. Honeypots are fake computer systems that allow the detection of unusual interactions. Honeytokens serve the same purpose but are anything but a computer. (de Barros, 2007) (de Barros, 2007) (Spitzner, 2003)

Regarding cyber deception for web services, multiple specific techniques can be implemented. In this

work, the following methods were considered, based on the work of (Fraunholz et al., 2018b):

- *Version Trickery* is a form of dissimulation that provides falsified version information. This could be the version of the HTTP Server, the Operating System, the CMS installation, plugins or themes. This makes it harder for the attacker to identify vulnerable versions or find exploits. In the context of CMSs, it is possible to obfuscate the version completely, as described in Section 3.3. For WP, existing plugins can fake the version, see Table 2.
- *Disallow Injection* refers to injecting *Disallow* entries in *robots exclusion file*. These entries are used by web crawlers as well as different scanners, such as CMS Scanners, to find possibly sensitive paths and files. This falls into the category of simulation and can be used for detection purposes. The WP plugin 'Blackhole for Bad Bots' (Starr, 2022) is an example for faking entries in the *robots* file.
- *Status Code Tampering* refers to manipulating the HTTP response codes and could be used against scanners relying on this status code to test whether a certain path exists or not. It was used in this work to hide and fake the existence of specific plugins and is further described in Section 3.1.
- *Latency Adaption* is a simulation tactic to prevent brute-force attacks and can be thought of as a veiled rate limiting. The more requests are sent, the more the response from the server will be delayed, which limits the number of requests that the attacker can send. Rate-limiting brute force login attacks is a method used by different CMS security plugins (Sec. 5).
- *Virtual Honey Files* is a method where invented resources on the website are presented, with the goal of creating uncertainty for the attacker. This could be a fake directory, document, login page, back end, CMS installation etc. In this work a similar method was used to include plugin version numbers as described in 3.1.
- *Code Obfuscation* is used to hide the functionality of code such as *JavaScript*. This is used to prevent the stealing of intellectual property. It might be effective against CMS scanners when specific patterns in the HTML are analysed to extract information, but has not been applied in this work.
- *Cookie Scrambling* can be used as either simulation or dissimulation tactic. This is done by either creating a honey cookie, where tampering with the cookie can be detected or by creating cookies that need to be set to successfully execute a

task. An example of this for Wp is described by a blog post (Talk, 2022), which is explained more closely in Section 5.

- *Content Modification* can be used for dissimulation and simulation. Example for content modification are described in Sections 3.3 and 3.4 where users and versions are hidden. Furthermore, as seen in Table 2, it can also be used to fake users and the version.

2.2 CMS

A Content Management System (CMS) is a way of managing content, which includes the collection and usage of the content. This allows for consistency and less repetition for the user. (Boiko, 2005, p. 72) There are two types of CMS: enterprise CMS, which contains content relevant to the whole organisation, and web CMS's, which focus is on managing content for websites. (Boiko, 2005, p. 79-82)

2.3 CMS Scanner

CMS scanners are a type of vulnerability scanner. Mainly, they enumerate a CMS to find as much information that can be used to determine vulnerabilities. There are different open-source scanners. Some scanners work for multiple different CMSs others are for a specific CMS.

In Table 1, a comparison was made between different CMS scanners and their functionalities. Module enumeration depends on the CMS and is generally done through directory brute forcing. For example, in WordPress, it would be plugin and theme enumeration. With one exception, every scanner has this functionality since these modules prone to contain vulnerabilities. Version detection shows similar results. User enumeration is possible for five out of the nine scanners. The detection of misconfigurations can be done by six scanners. URLs refer to the detection of 'interesting' URLs that are not necessarily searched for. Exposed files include backup, default, log and other files. Three scanners can do brute-force login attacks. Drupwn is the only scanner that has the integrated possibility of also exploiting found vulnerabilities, while some of the other scanners might link to possible vulnerabilities. Finally, WPSeKu allows for static code analysis, which no other scanner offers. (InfosecMatter, 2020)

2.4 WPScan

WordPress is the most popular, free and open-source CMS on the market. It allows the creation of web-

sites and blogs with high customizability through a wide range of community-created themes and plugins, which allow for customized style and functionality, respectively.

WPScan is one of the most known and advanced CMS scanners for WordPress with a large feature set (Tab. 1). In the work the focus is on its capabilities of theme, plugin, user and version enumeration and detection.

3 SCANTRAP WPScan EVADER

For evaluating the ability to evade CMS scanners, we used WordPress as the CMS and WPScan as the CMS scanner and created a plugin for WordPress that focused on the four main scans of WPScan. A the source code of WPScan is publicly available (WP-ScanTeam, 2022), it was used to better understand how the scans are performed and how the information is gathered.

Although WPScan reveals itself by setting the HTTP User-Agent field to 'WPScan', this information was not used, as it could be easily changed for malicious purposes. Instead, cyber deception was used, as explained in 2.1, to manipulate the scanner output. Additionally, we partly adapted and modified ideas from a blog post (Cyberpunk, 2019) on this matter.

3.1 Plugin Detection

Plugin detection is how WPScan finds existing plugins as well as their versions. The focal method for finding plugins is based on known locations. The WordPress structure has a determined folder ('/wp-content/plugins/') for plugins. Therefore WPScan does a directory scan for known plugins in this folder. If the directory for the plugin exists, which is based on the response code, then the mandatory 'readme.txt' file is requested and the version is parsed from either the 'Stable Tag' or 'ChangeLog Section'.

For this work, two methods to evade plugin detection were used. The first approach is based on simulation. We faked plugins by changing the 404 response code of requests to non-existing plugins. WPScan determines a plugin as existing, when one out of four response codes is detected. The code "200 OK" means the request succeeded. "401 Unauthorized" stands for unauthenticated access and "403 Forbidden" means the user has no access rights. Lastly, "500 Internal Server Error" indicates an error on the server's side. For responses to plugin folder requests, we chose the 403 response code and for the 'readme.txt' file we se-

Table 1: Comparing different CMS scanners functionality (InfosecMatter, 2020).

Scanner	CMS	Module Enumeration	Version Detection	User Enumeration	Misconfigs	URLs	Exposed Files	Login Attacks	Exploits	Code Analysis
Droopescan	multiple	✓	✓	-	-	✓	✓	-	-	-
CMSmap	multiple	✓	✓	✓	✓	✓	✓	✓	-	-
CMSeek	multiple	✓	✓	✓	✓	-	✓	-	-	-
WPScan	WordPress	✓	✓	✓	✓	✓	✓	✓	-	-
WPSeku	WordPress	✓	-	-	✓	-	✓	✓	-	✓
JoomScan	Joomla	✓	✓	-	✓	-	✓	-	-	-
JoomlaVS	Joomla	✓	✓	-	✓	-	-	-	-	-
JScanner	Joomla	-	✓	✓	-	-	-	-	-	-
Drupwn	Drupal	✓	✓	✓	-	-	✓	-	✓	-

lected status code 200. Further, we added customised text to the response body to add a stable tag for version detection. As an addition, based on honeytokens, we added simple logging when fake plugin paths are requested. This extends deception as a defence to also be able to detect unusual behaviour. An alternative to logging could have been an email notification.

The second method of hiding the existing plugins, is a method of dissimulation. The plugins are hidden by responding to requests to the plugin folder with a '404 Not Found' response. This is possible because WPScan will request files only when the folders are found. It still allows WordPress to use the contents of the folders, therefore not restricting the functionality of the plugins.

Figure 1 shows the sequence of request and response flow. It illustrates the manipulation of responses for plugin folder requests as described above.

3.2 Theme Detection

Theme detection is identical to plugin detection, only with a change of the folder ('/wp-content/themes/'). The version is detected through the 'style.css' file and the version tag.

Besides finding all installed themes, WPScan looks for the 'main theme'. The 'main theme' is the currently active theme that styles the page. To detect this theme, WPScan scans a website's content for links containing the themes folder.

To evade theme detection, we used the same two methods as for the plugins, as described in Section 3.1. This includes logging fake themes.

Evading the main theme detection would require changing links inside the website, which would lead to a loss of the theme style. Since this is a significant loss of functionality, we decided to not change it.

3.3 Version Detection

Version detection refers to detecting the running version of the WordPress installation. WP displays the version in different ways. WPScan can detect the following:

- *WordPress Head:* WordPress prints the version in the head of the website.
- *WordPress Generator:* WordPress adds the version the the XML generator.
- *Version Query:* WordPress also shows the version as a query argument for core style and script files.
- *Fingerprinting:* All core JavaScript, CSS and JSON files of a WordPress version have a unique hash with which the version can be detected.
- *File Content:* Both the 'install.php' and 'load.styles.php' scripts output the version.

To break the version detection, we removed the WP head and generator as well as the version query arguments. We added a space to all the core files to change the hashes, so fingerprinting is no longer possible. Finally, we removed the lines of code in the scripts, thus they no longer display the versions.

Regarding simulation, it should not be possible to manipulate the WordPress head or version query. In theory, it is possible to change the generator response. Fingerprinting could be simulated by adding core files from older versions. This is, however, not desirable since WordPress should be up to date. The php file content of the 'install.php' could be changed instead of removed, but the 'load.styles.php' appends versions to the links to get the correct file and therefore also could not be simulated.

3.4 User Enumeration

User enumeration tries to detect existing usernames to log into the WordPress instance. For this WPScan uses different functionality that WordPress offers:

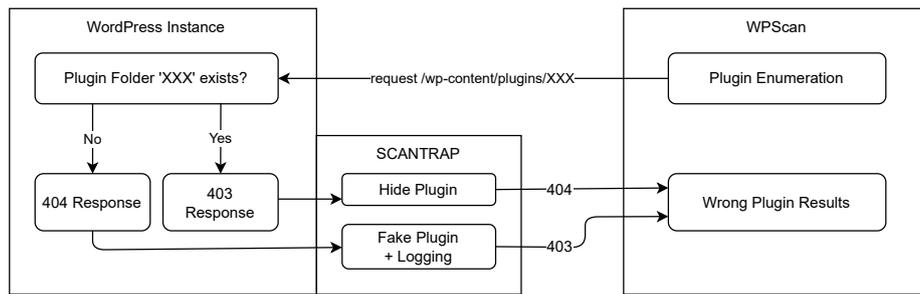


Figure 1: Demonstration of response manipulation of request for plugin folder 'XXX' with WP plugin 'SCANTRAP'.

- **REST API:** WordPress allows to query users through the available REST API.
- **JSON API:** The JSON API also makes it possible to query users.
- **Author Class:** When authors of posts answer comments, they are marked with an author class.
- **RSS Author Tag:** The RSS feed contains a specific tag naming the author.
- **Login Error:** Login error messages can display if the user exists.
- **URL Queries:** WordPress allows to do URL queries with the user id.
- **Author URL:** WordPress adds an author link to posts that contains the username.

Completely evading user enumeration comes with a loss of functionality. The REST and JSON API are disabled, which restricts the ability of applications to interact with your WordPress site. URL queries are forbidden. The author class, author link and RSS author tag are removed, which reduces the visibility of authors. All login errors are disabled, which might make troubleshooting more difficult. This could potentially also effect search engine optimization.

With the exception of the RSS Author Tag, all other areas can not be used for simulation purpose since it would restrict the functionality. It might be possible to create a user purposely as kind of a 'honeypot' user. However, the plugin removes all user enumeration. The RSS Author Tag though could be overwritten with an arbitrary name.

4 DISCUSSION

We tested our plugin in a test environment. Details and the WPScan output can be found in the Git repository (<https://github.com/dfki-insec/SCANTRAP>). The results show that the proposed strategies work for all four scans, with the exception

of the detection of the main theme, as described in Section 3.2.

One downside regarding our methodology is that some of the functionality, that might be desired by the users, are restricted with the use of our plugin. Especially user enumeration leads to potentially undesirable loss. However, user enumeration could be seen as more of a privacy concern. Every defender would need to decide for themselves if they put more importance on functionality or privacy. One exception to this assumption is that WordPress usernames are also login names and WordPress does not restrict login tries. This makes user enumeration interesting in the context of login brute force attacks. As mentioned in 3.2, the main theme detection is one example of full loss of the main functionality 'style'. However, it has to be said that themes only make up 6% of the existing vulnerabilities (WPScan, 2022). A similar problem could occur if plugins require URLs to the plugin folder on any pages. Another problem that could occur is with search engine optimization, since the plugin removes author links and manipulates status codes to certain folders and files.

We are aware of two limitations in this methodology: the first is our testing environment, as it is not a real WordPress blog. We tested with only a limited amount of plugins, themes, users and content. Due to a large number of existing plugins alone, it would not be feasible to test everything. The second is the restrictions on CMSs and CMS scanners. Our plugin is only focusing on WordPress and WPScan. As shown in Section 2 there are more CMSs and scanners that might require adjustment to our method.

Cyber deception is supposed to deter attackers and make attacking harder. It will not remove vulnerabilities or definitely hinder an attacker from targeting the CMS. Therefore, having other security measures, like up-to-date software, is still required. However, it might discourage attackers completely from seeing the CMS as a target. More likely, it will require more resources for additional manual reconnaissance or lead attackers to the wrong conclusion about exist-

ing vulnerabilities. This might be helpful as a defence method for overlooked vulnerabilities.

5 RELATED WORK

While research in regards to securing web content and CMSs has been conducted, we have found little in regards to using cyber deception, specifically for CMSs.

Jagamogan et al. (Jagamogan et al., 2021) conducted a review on different penetration testing approaches on CMS. It underlines the likelihood of CMSs being targeted. While the focus lies on exploitation frameworks, it is apparent that a lot are based on different scanning methods. Specifically, it is mentioned that WPScan and other CMS scanners were used for attacks.

The paper by Jerković et al. (Jerković et al., 2016) describes in more detail the problems and different vulnerabilities of CMSs. Such as easy CMS identification and a large base of unsecured and out-of-date plugins. For protection, the paper lists different best practices, such as content delivery networks (CDN), regular updating and logging. They conclude that an up-to-date CMS is not sufficient enough to prevent attacks. Their solution for this is hosting in CDN, which requires additional services and potential costs.

Efendi et al. (Efendi et al., 2019) highlighted the use of deception techniques in the domain of web applications. Their study describes web application vulnerabilities and attacks and deception techniques. Their focus lies on the detection of different attacks on the application layer. The concentration is on services and processes like SSH, Javascript and MySQL and attacks such as Denial of Service and SQL Injections. In comparison, we are not focusing on detecting specific attacks but rather hindrance of successful reconnaissance and detection of such in the specific domain of CMSs.

Valenza et al. (Valenza et al., 2020) developed a prototype that attacks vulnerability scanners. In their attacker model, the attacker becomes the victim and the defender the attacker. The idea is to use vulnerabilities of the vulnerability scanner application to attack it through HTTP responses. In regards to attacks, their focus is on XSS and exploitation to get access to the machine. Our paper is based on the same idea that vulnerability scanners can not be attacked or manipulated. Instead of finding vulnerabilities in scanners, however, our work focuses on finding structures to manipulate the outcome of the scanner's results.

A blog post from Medium Talk (Talk, 2022) describes two methods to hide the login URL. The first is to change the name of the login file, the second is

to use a cookie to control access to the login page. A similar approach to this is taken by the 'Hide My WP Ghost - Security Plugin' (Plugins, 2022). This plugin allows changing and hiding all major paths and files as well as some additional features.

The blog post from *Cyberpunk* (Cyberpunk, 2019) that we have built our plugin on, is to our knowledge one of a few comprehensive resources on using cyber deception for CMSs. The post focuses specifically and only on WordPress and WPScan. The first part of the blog describes the usage of WPScan. The second part goes into the defence against detection. For each area, it gives a short description of how WPScan finds information and a block of code that should prevent it. Some of the defence methods require changes outside of WordPress. The focus of the blog post lies in hiding information. We found the listings of sources used by WPScan incomplete and therefore, some of the code snippets did not lead to the desired effect on their own. We build upon this by adding defence against additional sources, removing the need to manually change settings in web servers and adding upon the idea by including additional cyber defence techniques like simulation and honeytokens.

Another blog post that deals with WPScan obfuscation is by *shift8* (Shift8, 2019). It deals specifically with blocking WPScan with Nginx. It has similar content and drawbacks to the previously described blog post. However, it additionally mentions the approach to blocking plugin and theme folders by changing response codes to 404 status codes.

The first posts on blocking web CMS scanners, we found, are about user enumeration (Andreasen, 2016) and blocking WPScan (McRae, 2017). Both posts can be considered partly outdated and incomplete. Both focus on dissimulation and do not include simulation.

We have found a few plugins in the WordPress store that have similar security and deception ideas. Table 2 gives an overview of the different functions offered by them. We are mainly looking at hiding and faking, so dissimulation and simulation respectively. Additionally, the detection and blocking of attackers is a broad term for whitelisting, blacklisting IPs, as well as preventing brute-force attacks. Some of the plugins might also include some additional functionality.

'Blackhole for Bad Bots' creates honeysites and blocks IP addresses (Starr, 2022). In a similar manner, 'tinyShield' (deprecated) adds functionality of white- and blacklisting (tinyShield.me, 2022), so does 'NovaSense' (NovaSense, 2020). Another plugin called 'BlogSafe HoneyPot' tracks failed URL requests and login attempts (BlogSafe.org, 2021). 'Astra Security Suite' contains IP blocking mechanisms, logging, at-

Table 2: Comparing different WordPress security plugins regarding their dissimulation and simulation features.

Plugin / Capability	Hide/Fake Users	Hide/Fake Plugins	Hide/Fake Themes	Hide/Fake 'wp-content'	Hide/Fake Login Path	Hide/Fake Login Error	Hide/Fake Version	Hide/Fake Cookies	Hide WordPress	Hide/Fake robots.txt	Detect/Block Attackers
SCANTRAP	✓/-	✓/✓	✓/✓	-/-	-/-	✓/-	✓/-	-/-	-	-/-	✓/-
Blackhole (Starr, 2022)	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-	-/✓	✓/✓
WP Ghost (Plugins, 2022)	✓/-	✓/-	✓/-	✓/-	✓/✓	-/-	-/-	-/-	✓	-/-	✓/✓
Don Security (Donini, 2017)	✓/-	✓/-	-/-	-/-	-/-	-/-	✓/-	-/-	-	✓/-	-/-
Stop User Enum (Fullworks, 2022)	✓/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-	-/-	-/-
WP Smart Security (WPSmartSecurity, 2015)	-/-	✓/-	✓/-	✓/-	✓/-	✓/-	✓/✓	-/-	✓	-/-	✓/✓
Titan (CreativeMotion, 2022)	-/-	-/-	-/-	-/-	✓/-	-/-	✓/-	-/-	-	-/-	✓/✓
tinyShield (deprecated) (tinyShield.me, 2022)	✓/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-	-/-	✓/✓
block-wpscan (rлуisr, 2016)	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-	-/-	✓/✓
NovaSense (NovaSense, 2020)	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-	-/-	✓/✓
Astra Security (AstraSecurity, 2020)	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-	-/-	✓/✓
BlogSafe HoneyPot (BlogSafe.org, 2021)	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-/-	-	-/-	✓/-

tack prevention and more (AstraSecurity, 2020). This is similar to the plugin 'WP Smart Security', however, it has not been updated in a while (WPSmartSecurity, 2015). The 'Titan Anti-spam & Security' plugin does spam detection, has a login rate limiter and uses WP-Scan (CreativeMotion, 2022). The WPScan database is also used by 'Jetpack Protect' (Automatic, 2022). Now, all of these plugins have their own technique for securing WordPress with a strong focus on attack prevention and less on scanner deception.

We found three plugins that focus more on scanner evasion with similar methods to our plugin. 'Stop User Enumeration' focuses only on user enumeration (Fullworks, 2022). The plugin 'block-wpscan' is based on detecting browser languages and user agents, however, it is no longer maintained and WP-Scan can evade this type of detection (rлуisr, 2016). Finally, the plugin 'Don Security' (Donini, 2017) seems to be closest to our plugin. Its goal is to prevent user enumeration, version detection and plugin detection, specifically for WPScan. It seems to be based on some of the mentioned outdated blog posts and is also no longer maintained.

6 CONCLUSIONS

In this paper, we have presented a WordPress plugin that manipulates the output of WPScan scans, in order to counter reconnaissance. This plugin can hide the existence of plugins, themes and users from the enumeration process and present false plugins and themes. The results support the idea that cyber deception and obfuscation methods are applicable to the CMS enumeration process, and can also help to detect attacks at an early stage. This could be effective to protect from 'script kiddies', bots or attackers that use automation to gather information. Future work could generalize this approach and transfer it to other CMS than WordPress. It should be noted that scanning tools could be adapted to be more resilient against cyber deception, representing a cat-and-mouse game.

ACKNOWLEDGEMENTS

This research was supported by the German Federal Ministry of Education and Research (BMBF) through the Open6GHub project (Grant 16KISK003K).

REFERENCES

- Andreasen, M. (2016). Protect + Stop WPScan WordPress User Enumeration with Varnish. <https://guides.wp-bullet.com/protect-stop-wpscan-wordpress-user-enumeration-varnish/>. Accessed: 2022-09-28.
- AstraSecurity (2020). Astra Security Suite – Firewall & Malware Scan. <https://wordpress.org/plugins/getastra/>. Accessed: 2022-09-28.
- Automatic (2022). Jetpack Protect. <https://wordpress.org/plugins/jetpack-protect/>. Accessed: 2022-09-28.
- Behera, C. K. and Bhaskari, D. L. (2015). Different Obfuscation Techniques for Code Protection. *Procedia Computer Science*, 70:757–763. Proceedings of the 4th International Conference on Eco-friendly Computing and Communication Systems.
- Bell, J. B. and Whaley, B. (1991). Cheating and Deception (1st ed.). *Transaction Publ, New Brunswick*.
- BlogSafe.org (2021). BlogSafe HoneyPot. <https://wordpress.org/plugins/blogsafe-honeypot/>. Accessed: 2022-09-28.
- Boiko, B. (2005). *Content Management Bible*. Bible. Wiley.
- CreativeMotion (2022). Titan Anti-spam & Security. <https://wordpress.org/plugins/anti-spam/>. Accessed: 2022-09-28.
- Cyberpunk (2019). WPScan Usage Example [Enumeration + Exploit]. <https://www.cyberpunk.rs/wpscan-usage-example>. Accessed: 2022-09-21.
- de Barros, A. (2007). Dlp and honeytokens.
- Donini, R. (2017). Don Security. <https://wordpress.org/plugins/don-security/>. Accessed: 2022-09-28.
- Efendi, A. M., Ibrahim, Z., Zawawi, M. A., Abdul Rahim, F., Pahari, N. M., and Ismail, A. (2019). A Survey on Deception Techniques for Securing Web Application. In *2019 IEEE 5th Intl Conference on Big Data Security on Cloud*, pages 328–331.
- Fraunholz, D., Anton, S. D., Lipps, C., Reti, D., Krohmer, D., Pohl, F., Tammen, M., and Schotten, H. D. (2018a). Demystifying Deception Technology: A Survey.
- Fraunholz, D., Reti, D., Duque Anton, S., and Schotten, H. D. (2018b). Cloxy: A Context-Aware Deception-as-a-Service Reverse Proxy for Web Services. In *Proceedings of the 5th ACM Workshop on Moving Target Defense*, MTD '18, page 40–47. ACM.
- Fullworks (2022). Stop User Enumeration. <https://wordpress.org/plugins/stop-user-enumeration/>. Accessed: 2022-09-28.
- InfosecMatter (2020). CMS Vulnerability Scanners for WordPress, Joomla, Drupal, Moodle, Typo3.. <https://www.infosecmatter.com/cms-vulnerability-scanners-for-wordpress-joomla-drupal-moodle-typo3/>. Accessed: 2022-09-21.
- Jagamogan, R. S., Ismail, S. A., Hafizah, N., and Hafiza Abas, H. (2021). A Review: Penetration Testing Approaches on Content Management System (CMS). In *2021 7th International Conference on Research and Innovation in Information Systems (ICRIIS)*, pages 1–6.
- Jerković, H., Vranešić, P., and Dadić, S. (2016). Securing web content and services in open source content management systems. In *2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1402–1407.
- McRae, S. (2017). Blocking the scanners — WP-Scan. <https://gorgolak.wordpress.com/2017/05/27/blocking-the-scanners-wpscan/>. Accessed: 2022-09-28.
- NovaSense (2020). NovaSense. <https://wordpress.org/plugins/novasense-security/>. Accessed: 2022-09-28.
- Plugins, W. W. S. (2022). Hide My WP Ghost – Security Plugin. <https://de.wordpress.org/plugins/hide-my-wp/>. Accessed: 2022-10-05.
- rluisr (2016). block-wpscan. <https://wordpress.org/plugins/block-wpscan/>. Accessed: 2022-09-28.
- Shift8 (2019). How to block your WordPress site from being scanned by WPScan with Nginx. <https://shift8web.ca/2019/01/how-to-block-your-wordpress-site-from-being-scanned-by-wpscan-with-nginx/>. Accessed: 2022-09-28.
- Spitzner, L. (2003). The honeynet project: trapping the hackers. *IEEE Security & Privacy*, 1(2):15–23.
- Starr, J. (2022). Blackhole for Bad Bots. <https://wordpress.org/plugins/blackhole-bad-bots/>. Accessed: 2022-09-28.
- Talk, M. (2022). How to Change The WordPress Login URL Without Plugin. <https://www.mediumtalk.net/change-wordpress-login-url-without-plugin/>. Accessed: 2022-10-05.
- tinyShield.me (2022). tinyShield – Simple. Focused. Security. <https://wordpress.org/plugins/tinyshield/>. Accessed: 2022-09-28 and 2022-12-19.
- Valenza, A., Costa, G., and Armando, A. (2020). Never Trust Your Victim: Weaponizing Vulnerabilities in Security Scanners.
- W3Techs (2022). Usage statistics of content management systems. https://w3techs.com/technologies/overview/content_management. Accessed: 2022-09-21.
- WPScan (2022). WordPress Vulnerability Statistics. <https://wpscan.com/statistics>. Accessed: 2022-09-20.
- WPScanTeam (2022). WPScanTeam WPScan Github. <https://github.com/wpscanteam/wpscan>. Accessed: 2022-09-21.
- WPSmartSecurity (2015). WP Smart Security. <https://wordpress.org/plugins/wp-smart-security/>. Accessed: 2022-09-28.
- Yuill, J. J. et al. (2007). Defensive Computer-Security Deception Operations: Processes, Principles and Techniques. <https://repository.lib.ncsu.edu/handle/1840.16/5648>. Accessed: 2022-09-21.