

# **A Novel WordPress Plugin *WP KeepSoul* To Detect And Mitigate Malicious Code And Maintain Business Continuity In The WordPress Platform**

**DETAILED PROJECT PAPER**



Author and Developer

PRANEETHRAJ

**2022-2023**

# CONTENTS

	Page No
<b>CHAPTER 1. ABSTRACT</b>	<b>1</b>
<b>CHAPTER 2. INTRODUCTION</b>	<b>2</b>
2.1 Aims, Objectives and Contributions	3
2.1.1 Research Aim	3
2.1.2 Research Objectives	3
<b>CHAPTER 3. LITERATURE REVIEW</b>	<b>5</b>
3.1 WordPress Vulnerabilities and Attack Vectors	5
3.2 Existing Malware Detection Methods and Obfuscated JavaScript Detection	7
3.3 Malicious Keywords Extracted for Implementation as Detection Keywords and Characters.	9
3.4 Analysis of existing solutions for understanding knowledge gaps	9
3.5 How the Literature Review Helps to Curate the Artefact	10
<b>CHAPTER 4. METHODOLOGY</b>	<b>12</b>
4.1 Ethical Disclosure	13
4.2 Testing Environment and Dataset	13
4.3 Classification Of Malware Used for Testing	13
4.4 Modules Developed and Their Functions	17
4.4.1 Module 1: Activation Module; download WordPress from GitHub and create necessary files and folders for operation.	17
4.4.2 Module 2: Full Scan Module uses the PHP file comparison function to detect malicious files and restore legitimate files.	18
4.4.3 Module 3: Quick Scan Module; use keywords and characters frequency set to detect malicious files and restore legitimate files.	19
4.4.4 Module 3.1: List of keywords maintained in a file that feeds quick scan detection (rules.php).	20
4.4.4 a) Algorithm of the quick scan process	21
4.4.5 Module 4: Log Module; To record all full and quick scan incidents.	21
4.4.6 Module 5: Email Module; PHP mail module to send the administrator quarantined files and log files.	22
4.6 Building Attack Simulation	24
4.7 Attack Simulation	25
4.7.1 First Method: Injection Script Working	25
4.7.2 Second Method: Manually place specific malware	26
<b>CHAPTER 5. TEST RESULTS</b>	<b>27</b>
5.1 Malware Test Results	27
5.1.1 Result Comparison Criteria	27

5.1.2 Effects of Malware Before Scan	28
5.2 Module Test Results	30
5.2.1 Module 1: Activation Module	30
5.2.2 Module 2: Full Scan Module	32
5.2.3 Module 3: Quick Scan Module	33
5.2.4 Module 4: Log Module	34
5.2.5 Module 5: Email Module	35
<b>5.1.7 Artefact in Action</b>	<b>37</b>
<b>CHAPTER 6. EVALUATION AND DISCUSSION</b>	<b>40</b>
6.1 Results Evaluation and Limitations Based on Modules	40
6.1.1 Module 1: Activation Module	40
6.1.2 Module 2, 3 & 3.1: Full Scan and Quick Scan Module	40
6.1.3 Module 4: Log Module	42
6.1.4 Module 5: Email Module	42
<b>CHAPTER 7. CONCLUSION AND FUTURE WORK</b>	<b>43</b>
<b>CHAPTER 8. APPENDIX LIST</b>	<b>44</b>
A. Installation Instruction	44
B. Source Code Link: <a href="https://gitlab.uwe.ac.uk/p2-praneethraj/wp-keepsoul">https://gitlab.uwe.ac.uk/p2-praneethraj/wp-keepsoul</a>	44
C. Walkthrough Video Link: <a href="https://youtu.be/E_WOkIVnIH8">https://youtu.be/E_WOkIVnIH8</a>	44
D. Malware scan report - filtered as full scan with malware classification	44
E. Malware scan report - filtered for quick scan with malware classification.	44
F. Record of supervisor meetings. schedule and notes.	44
G. Scan log from the testing environment - Combined full scan & quick scan.	44
<b>CHAPTER 9. REFERENCES</b>	<b>45</b>

## LIST OF TABLES

Table 01. Comparing the existing similar plugins and their capabilities is questioned to understand the gap	10
Table 02. List of characters and keywords shortlisted to implement in the quick scan.	11
Table 03. Classified malware based on its use and named based on date as mentioned in (Pejcic, 2022)	14
Table 04. Selected Malware that created critical issues in WordPress history	15
Table 05. Full Scan results, Limited (Complete list in Appendix D)	32
Table 06. Quick Scan Results	34
Table 07. Entry of four different logging cases	34

PRANEETHRAJ WP KEEPSOUL Version 1.0

# LIST OF FIGURES AND FLOWCHART

Figure 01. The idea of adversary acts on website and server.	16
Figure 02. Code snippet for downloading a new copy of the installed version.	17
Figure 03. Code snippet for comparing existing and new copies of files.	18
Figure 04. Code snippet of rules.PHP file	21
Figure 05. Code snippet used to generate log.csv file.	21
Figure 06. wp-mail() - a PHP mailing code snippet from the WordPress development repo.	22
Figure 07. Time taken by different plugins for different metrics, as mentioned.	23
Figure 08. Code snippet from inject.PHP script used.	25
Figure 09. 28.01.2021 - index.PHP malware effects on the testing website.	28
Figure 10. 09.03.2022 worker.PHP malware effects on the testing website.	28
Figure 11. 08.10.2021 malware effects on the testing website.	29
Figure 12. 18.05.2021 malware effects on the testing website.	29
Figure 13. 04.02.2022 - malware effects on the testing website.	30
Figure 14. Creation of directories and downloading WordPress.	31
Figure 15. Creation of directories and downloading WordPress.	31
Figure 16. Snapshot of logs recorded.	35
Figure 17. Email received after plugin activation.	36
Figure 18. Email received after a full scan.	36
Figure 19. WP KeepSoul after activation in dashboard page	37
Figure 20. WordPress Dashboard Settings panel with plugin access.	37
Figure 21. Plugin Page, with Instructions and option to perform a scan and download log.	38
Figure 22. Notification after completing Quick Scan with the time taken.	39
Figure 23. Notification after completing Full Scan with the time taken.	39
Figure 24. Snapshot of a full scan, representing the average time of the full scan.	41
Figure 25. Snapshot of scan logs.	42
Flowchart 01. Flowchart of the process flow of the full scan module.	19
Flowchart 02. Flowchart of the process flow of the quick scan module.	20

## LIST OF ABBREVIATIONS

URL	Uniform Resource Locator
SMART	Specific, Measurable, Achievable, Relevant, Time-bound
URI	Uniform Resource Identifier
DDOS	Distributed denial of service
VPS	Virtual Private Server
CDN	Content Delivery Network
API	Application Programming Interface
GUI	Graphical User Interface
PHP	Hypertext Preprocessor (a programming language)
CMS	Content Management System
SEO	Search Engine Optimisation
CVSS	Common Vulnerability Scoring System
XSS	Cross-Site Scripting
OWASP	Open Web Application Security Project
TAC	Theme Authenticity Checker
SQL	Structured Query Language
CPU	Central Processing Unit
RAM	Random Access Memory
APT	Advanced Persistent Threat
HTTP	Hypertext Transfer Protocol

# CHAPTER 1. ABSTRACT

Threat actors may deliberately inject and upload malicious scripts and files to take over web applications and servers. In web applications, malicious scripts often come with novel obfuscation techniques; therefore, it becomes difficult to match the signatures and detect malware, even if the underlying malicious script(s) had been identified earlier. Sucuri highlights that, out of all content management systems used, WordPress was the most targeted. The statistics indicate that 61.65% of malware redirects users to some malicious site or steals credit card information from websites by injecting malicious code (Garand et al., 2021). To solve this issue, we proposed a novel WordPress plugin WP-KeepSoul that aligns with the rule of secure software development from Cybok: "*Keep the design of the system as simple and small as possible*" (Williams, Massacci and Gary, 2021). The proposed solution detects any malicious code in the core files and removes arbitrarily uploaded files. Additionally, an automated email system sends a complete log report and a zip of the quarantined files to the site administrator.

We use more than 3,000 samples of JavaScript malware from HynekPetra's git repository (Petra, 2022) and 91 Hypertext Preprocessor (PHP) malware from Pejic's git repository (Pejic, 2022) to test the artefact and compare results against the existing state-of-the-art solutions. The outcome was that WP-KeepSoul stood out in the detection with only three errors. The time required to detect and bring the website to a clean state was just 9.27 seconds, which was significantly quicker than existing solutions. The literature review aimed to identify current issues, analyse existing solutions, and identify potential knowledge gaps by answering relevant technical questions. We follow a module-wise structure across the report, and a quantitative analysis of plugin performance and hit-and-miss scan rate with time efficiency is discussed.

Finally, a critical evaluation of achieved aims and objectives and their outcome with the limitations are discussed. This evaluation gives a clear pathway to improving the artefact, enhancing efficiency in different dimensions.

**Keywords:** anti-malware, WordPress, plugin, version-independent solution, simple and quick recovery.

## CHAPTER 2. INTRODUCTION

The website is the company's digital face, representing its identity with information like who it is, what it does, and who its clients and customers are. Many companies regularly use their web applications to carry out primary business operations. WordPress is a popular open-source Content Management System (CMS), and 65% of all websites use WordPress as a framework (W3Techs, n.d.). WordPress allows one to create various web applications such as organisation portfolio sites, online stores, business websites, portfolios, podcasts, online marketplaces and even e-learning platforms (Tuca, 2022).

In this chapter, we will first provide an overview of the background and main idea of the study. We will then present our research using a Specific, Measurable, Achievable, Relevant, Time-bound (SMART) structured approach. Next, our research goals, objectives, and questions will be outlined, along with the purpose of the study. Lastly, we will address any limitations of the research.

During a cyber attack, websites may experience defamation, long-term service disruptions, and potential theft of sensitive client and customer information, leading to significant financial and reputational harm. Although companies and developers try to implement many existing research solutions and use anti-malware plugins to protect their websites, around 4.7 million websites are hacked annually, an estimated 13,000 per day (Moran, 2022).

The research question for this study is to develop a solution using PHP language to protect a website from malicious file upload or script injection attacks by automatically fixing it with minimal downtime. It is essential to address the current manual process of comparing malicious files with original files and replacing them or using automated methods to restore the complete backup to a previous state can be onerous, involve many tasks and configurations, and result in extended downtime for larger websites.

Limitations are inevitable, and research takes more time than evolution in technology. Hence, to develop the methodology, the research papers about the latest WordPress plugins were bare minimum and the time required to implement email and the quick scan modules in methodology was inadequate. Despite it, the output justifies the requirement and requires further research and improvement.



Finally, the structure of the report follows an in-depth *literature review* of WordPress and the dimensions of malware, including finding the proper and specific technical gap in existing solution providers, followed by the *methodology*, which demonstrates the development artefact based on five different modules and performance analysis. Then comes the testing phase, where the *results* of attack simulation and detection with the quantitative analysis are described. Finally, evaluation and discussion comprise a critical review of the developed artefact, a discussion of its use with limitations, and in the end, *future work* and relevant *references* are mentioned.

## **2.1 Aims, Objectives and Contributions**

The primary focus of the dissertation, known as the research aim, is discussed in this section. Additionally, the research objectives, which depict the intended specific outcome and achievement of the research, are also outlined in this section. A novel method to solve the existing problems is highlighted, and the contribution of this research in providing a simple and easy solution is highlighted.

### **2.1.1 Research Aim**

The aim is to create a plugin for the WordPress platform that addresses and reports the presence of malware and misconfigured files. This plugin will assist website developers and administrators in efficiently recovering infected websites and minimising the time and effort required for recovery.

### **2.1.2 Research Objectives**

The research objectives cover literature study, gap analysis, developing artefact, analysing developed plugins, and testing with various malware. Major bullet points are listed below, covering all the aspects of the project:

- To determine the most common types of malware currently targeting WordPress websites.
- To evaluate the effectiveness of existing anti-malware solutions and identify potential gaps or areas for improvement by literature survey.
- To design and develop an anti-malware plugin using PHP that can effectively detect and remove a wide range of malware threats without using a Content Delivery Network

(CDN), Application Programming Interface (API) or malware signatures with the least detection time.

- To test and validate the performance of the developed antimalware plugin through the p3-plugin profiler tool and simulate the real-world attack on a WordPress website to get plugin detection results.
- To evaluate the security of the developed plugin and to investigate the efficiency of detection and the time taken.
- To document the findings, critically evaluate the result, and then provide knowledge about areas of improvement in the developed artefact.

### **2.1.3 Project Contribution**

In this academic project, we introduce a novel approach for detecting file or script injection malware effectively; it also detects missing core files and deletes any additional files maliciously uploaded. Additionally, our system can restore only a specific infected file rather than the entire backup, minimising the impact on the system resources and maintaining better business continuity.

One of the critical features of our method is that it does not require malware signatures or supporting API, which makes the system more efficient and cost-effective. Our approach also does not require a malware signature for the full scan module, making it even more efficient. With all of the features mentioned above, our approach significantly improves the security of WordPress websites and helps various stakeholders of development and cybersecurity teams to understand and mitigate issues.

## CHAPTER 3. LITERATURE REVIEW

A critical literature study was conducted to understand and create a base for our study and research, including peer-reviewed papers, technical papers, whitepapers, and WordPress plugin documentation. The study is divided into three main contents and context in mind, i.e., *WordPress vulnerabilities, existing malware detection methods, obfuscated JavaScript detection methods, and existing solutions analysis with test questions*. The overall takeaway of the current method's advantages and disadvantages and how the literature review helps curate the artefact is discussed at the end of the section.

### 3.1 WordPress Vulnerabilities and Attack Vectors

The OWASP is a dependable resource for understanding web vulnerability research. The article understanding file upload discusses risk factors, bypass and protection methods, and prevention methods. Incidents may involve total system compromise, excessive use of database resources or disk space, forwarding attacks to back-end systems, client-side attacks, or web defacement to damage the reputation (OWASP et al., 2020). The core files of a web application are an essential component that is targeted in many types of attacks mentioned above. By protecting these files, the potential damage caused by vulnerabilities can be minimised.

According to the report of Sucuri from 2022, WordPress is the most targeted CMS, with 61.65% of malware causing URL infection that can redirect users to malicious sites or steal credit card information on e-commerce sites. Additionally, 60.04% of attacks resulted in a backdoor that allowed unauthorised access to the dashboard through secret channels, compromising the environment. The third most common type of attack was Search Engine Optimisation (SEO) spam, which affected 52.60% of sites by setting up redirects, publishing spam posts, and inserting links to improve search engine optimisation and direct traffic to third-party websites (Garand et al., 2021). All of these issues have file or script injection as their starting point, which is the focus of our research. The report also emphasises the promotion of GoDaddy products as a solution to these problems; using them is out of the scope of our research.

According to a 2021 survey by Patchstack, there is a significant concern about file upload vulnerabilities, which our artefacts aim to remedy. The 2021 study was more thorough and found a 150% increase in vulnerabilities that popular plugin developers never patched. Analysing

usage statistics, we discovered numerous unknown attacks related to file upload continued even after updates and patches. Additionally, it became difficult to remove vulnerable plugins because they had direct dependencies on the website and database. Out of all the vulnerabilities in the report, file upload was the main entry point for various attacks and had a Common Vulnerability Scoring System (CVSS) value of 10; as it is a critical issue, addressing file upload vulnerability is a major focus of our project. Furthermore, a study of 58,000 popular plugins revealed that 98.22% of the vulnerabilities stemmed from plugins and external code rather than WordPress. Surveys of web developers and security managers, including internal employees and freelancers, show that 70% express concern about web security due to the proliferation of vulnerabilities (Patchstack, 2021). Overall, this article focused on the problem of vulnerabilities, which helped us comprehensively understand various vulnerabilities and user requirements.

Few studies exist on hacked WordPress websites, and the available research is limited. A case study from *WP HackedHelp* discusses the effects of spam link injection on WordPress websites (Expert, 2020). This malicious activity is initiated by hackers to gain access to a website, such as through reverse shells, spam keywords, inserting malicious hyperlinks, spamming customer databases, creating defamatory content, and displaying hackers' ad banners. The case study points out two main ways hackers exploit WordPress vulnerabilities: SQL injection and incorrect folder permissions. These vulnerabilities allow spam files and links to enter WordPress themes, plugins, and core files. The authors recommend using the Theme Authenticity Checker (TAC) plugin but note that the TAC plugin is outdated and deprecated. They also suggest using an exploit scanner, which may not detect the latest or zero-day attacks; this information helps to identify weaknesses and dependencies in WordPress security.

WP Hackedhelp website states that many detection techniques for identifying and removing malicious code in WordPress require expert knowledge. For example, using nulled themes or cracked WordPress themes that do not require activation keys increases the risk of infection as these themes do not receive updates and may contain malicious code. Signs of a hacked website include Google warning messages, .htaccess pirate, pop-ups and constant crashes. The article (Expert, 2018a) recommends some best practices, but these options are inefficient once a website has been hacked.

Portswigger, a web application security software company, published an article discussing the mechanisms and impacts of file upload vulnerabilities. The article states that these

vulnerabilities can arise due to developer oversight, exploitation of dependencies, and errors in executable file handling that can expose sensitive information, such as source code. Additionally, the article notes that flawed file type validation and unrestricted file upload can be exploited and suggests implementing defence mechanisms and patches as solutions (Portswigger, n.d.). However, these solutions are often incomplete, may require multiple fixes to address a single issue, and may not apply to older versions.

According to the SANS Institute's report, several mainstream concepts for securing file uploads exist. These include creating a new file with a numerical identifier, setting a maximum file size limit for uploads, allowing the uploading of zip or compressed files to prevent the extraction of large files on the server, converting file types to strip out malicious features, implementing file permission control, adding authentication or allowing only authenticated users to upload files, and limiting the number of files that can be uploaded (Ullrich, 2009). These ideas provided insight into the security features that should be included in our artefacts.

In 2012, author Dalili identified several significant issues related to file uploads, including altering core functionality, bypassing protections, making websites vulnerable, causing a denial of service, and listing the most vulnerable files for attacks (such as .htaccess, web.config, crossdomain.xml, clientaccesspolicy.xml, global.asa, and global.asax). Even though this report is from 2012, certain exploits, such as the FCKeditor bypassed anti-malware protection by uploading a .htaccess file and running a shell inside, remain relevant and can be found on legacy websites (Dalili, 2012). While this information is still useful for identifying sensitive files that can be implemented in the quick scan module of our artefact, the patches mentioned in the report are outdated; and cannot be used to fix recent file upload vulnerabilities.

### **3.2 Existing Malware Detection Methods and Obfuscated JavaScript Detection**

A promising browser-based JavaScript malware detection and prevention method presented by Livshits, Zorn and Seifert in 2010 describe a solution called Zozzle uses the abstract syntax of JavaScript structure features fed to a Bayesian classifier. The report shows that this approach has a low number of false positives and can detect malware quickly (Livshits, Zorn and Seifert, 2010); however, it requires extensive training, and its dataset may change over time, leading to decreased detection quality year after year.

In 2020, Kasturi and a team of researchers developed YODA, a solution that combines key elements such as metadata analysis of website backups and codes analysis (both syntactic and semantic) to understand and identify malicious plugins. While metadata and plugin source code analysis may be faster, the combination of malicious activity detection involves complex mathematical equations that were only validated with 120 websites from 2008 (Kasturi *et al.*, 2022). This sample size is outdated and small, making it difficult to verify the effectiveness of the YODA artefact. The study also showed that static websites are more susceptible to attacks; however, since WordPress and its legitimate plugins release patches and fixes frequently, there is no need to analyse website backups from 2008. Additionally, metadata analysis may yield inaccurate results today as the code and dataset for plugins change with updates and customisation by the user according to the website's requirements.

Detection of obfuscated malicious JavaScript code, presented by Alazab *et al.* in 2022, involves a feature extraction process followed by three stages of training and testing the model. Based on their study, 170 different features were extracted and tested. They also compared JSOD to the state-of-the-art approaches, Zozzle and Nofus, for detecting obfuscated benign and malicious scripts (Alazab *et al.*, 2022). The experimental results showed that JSOD could detect obfuscated scripts and their sophisticated variations. Conversely, the dataset was imbalanced, so the authors used an independent classifier approach to balance it. This paper supports the list of features extracted, which will help add keywords for the "quick scan" module planned in our project.

In 2018, He, Xu, and Cha proposed a combined technique for detecting malicious JavaScript code that combines static and dynamic analysis. The static analysis is based on obfuscated code, URL, functional behaviour, and character-based approaches, which are strategically classified and counted using an occurrence-based approach. The dynamic analysis extracts feature based on function invocation, data manipulation, and code structure and use classification-based models for detection (He, Xu and Cha, 2018). The authors used the Random Forest classifier to create the malicious JavaScript code detection model. This approach demonstrated the ability to detect 100% of malicious scripts. Despite this, the implementation could be more efficient, and it helped to understand more about JavaScript obfuscation, which is an input for the scanning method.

A technique using n-gram features extracted from an abstract syntax tree and a balanced dataset to detect malicious patterns through frequency testing with a random classifier is

presented in Fass et al.'s research in 2018. While this approach achieved a high detection rate for suspicious scripts based on frequency and number of values, it may not be effective for detecting short, simple malicious codes; it may result in false positives for shorter, less structured codes (Fass *et al.*, 2018).

A solution named SAISAN, presented by Md et al. in 2017, detects content injection vulnerabilities. Their approach is only effective for versions 4.7.0 and 4.7.1 of WordPress; currently, this solution is not feasible in real time. Conversely, there is no way to modify the artefact for later versions as there was a major code update in WordPress core files after WordPress version 5.2 (Md *et al.*, 2017). Despite this, similar vulnerabilities can still be present due to specific plugins and cause similar damage. This report helped to understand the implementation of solutions effectively to solve a broader range of problems.

### **3.3 Malicious Keywords Extracted for Implementation as Detection Keywords and Characters.**

In the study by Moog et al. in 2021, the authors delve into the techniques used to hide malicious content through specific keywords and strings. Six code transformation techniques are identified: randomisation, data obfuscation, logic structure obfuscation, dynamic code generation, environment interaction, and code protection. The paper also discusses various code transformation tools, such as obfuscator.io, jsfuck, gnirts, custom encoding, and js minifier, that can be used to obfuscate code in different ways. Our main takeaway from this research is a deeper understanding of obfuscation techniques and the identification of commonly used keywords.

### **3.4 Analysis of existing solutions for understanding knowledge gaps**

Investigation of widely used anti-malware was conducted and determined some technical gaps that still need to be addressed in specific points, such as functionality, modules used, and user reviews. Our gap analysis highlights the specific questions to develop a practical solution, as shown in Table 01. Consequently, the development of WP-KeepSoul is being undertaken with the inclusion of attributes that are not present in other widely used plugins but are highly sought after by many users globally.

Question to Understand Knowledge Gap	Sucuri	Word fence	Malcare	iThemes	Clean Talk	WP Vivid	Astra Security	Backup Guard Security	Bullet Proof Security
1) Do plugin find malicious and misconfigured WordPress files?	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
2) Are there real-time detection and prevention methods?	No	No	No	No	Yes	No	No	Yes	No
3) Will the plugin only replace the affected file from the backup?	No	Yes	Yes	No	No	No	Yes	No	Yes
4) Will this plugin keep a backup of the infected files?	Yes	No	No	No	No	No	No	No	No
5) Does this plugin detect file permission changes and rewrite them with the necessary ones?	No	No	No	No	No	No	Yes	No	Yes
6) Will this work without downtime?	No	Yes	Yes	No	No.	No	Yes	No	No
7) Is there any email that notifies of website downtime?	No	Yes	No	No	No	No	No	No	No
8) Does it run cron jobs for detection?	No	No	No	No	No	Yes	Yes	Yes	Yes
9) Does the plugin send infected file to quarantine or save it for administrator investigation?	No	No	No	No	No	No	No	No	No
10) Is there any logs maintained for every particular file and scan?	Yes	Yes	Yes	No	No	No	No	No	No

**Table 01. Comparison of the existing similar plugins and their capabilities questioned to understand the knowledge gap.**

### 3.5 How the Literature Review Helps to Curate the Artefact

The literature review provided inputs that revealed many extant vulnerabilities and deficiencies. These inputs illuminated the understanding of issues that still need to be addressed by many anti-malware plugins, detection and prevention systems. Literature reading also helped to question, “what if there is a solution which will detect these malicious activities irrespective of the domain and environment? which platform is the number one target from attackers? Why are many studies based on Abstract Syntax Tree (AST) rather than direct keywords?” Moreover, this thought helped to explore different detection methods, JavaScript malware, and obfuscation techniques, which are discussed further in this section.



A comprehensive literature review was undertaken to identify the essential functions and keywords commonly used in malicious code, drawing on a range of sources, including Alazab et al. (2022), Fang et al. (2020), Patil and Patil (2017), Shah (2016), Fass, Backes, and Stock (2019), and Gupta and Gupta (2016, 2016b). The findings of this review are presented in Table 2, which lists the keywords and characters that will be used for detecting malware in a quick scan.

%		eval	setTimeout	substring
\$	\	charAt	iframe	replace
£	/	unescape	escape	document.addEventListener
"	?	while	classid	attachEvent
!	[	write	finally	createElement
&	]	getTempFilePath	fromCharCode	getElementById
*	{	saveToTemp	ActiveXObject	document.write
(	}	console.log	concat	search
)	~	do	indexOf	onload
onerror	.js	substr	String.charAt	location.assign
onunload	.php	parseInt	String.split	location.replace
onbeforeload	var	split	String.fromCharCode	getAppName
onmouseover	Math.random	strtonum	String.charCodeAt	getUserAgent
dispatchEvent	CharCodeAt	match	window.setInterval	createXMLHttpRequest
fireEvent	WScript	String.indexOf	window.setTimeout	dateObject.toGMTString
setAttribute	decode	objStream.Close	document.writeln	document.createElement
window.location	toString	getDataFromUrl	element.appendChild	chars
getCookie	setCookie	getData	element.innerHTML	&#x

**Table 02.** List of characters and keywords shortlisted to implement in the quick scan.

## CHAPTER 4. METHODOLOGY

From the literature and analysing the current requirement in the market, a solution is developed to cover most aspects to fill the existing gap in the website protection field. The core functionality of the artefact is to detect the malicious files using files from the backup to match the existing copy by character level comparison and keywords from Table 2 for the character-based match rather than an abstract syntax tree. Moreover, bifurcating the whole structure into two parts, a quick scan and a full scan for detection, can help to identify the issues faster. Also, the complete artefact is divided into five parts, modules one to five format is followed throughout all the phases of methodology, testing, results, and evaluation for better user readability and function understanding. This idea of artefact, i.e. retaining core applications files and comparing, can be implemented for platforms like Joomla, Drupal, Magento and other content management systems. **However, in order to provide a solution for the wider community, we implemented a plug-in solution for the WordPress platform.**

The necessary modules for the artefact, along with their preliminary tasks, are outlined as follows:

**Module 1:** Activation Module; download WordPress from GitHub and create necessary files and folders for operation.

**Module 2:** Full scan module; uses the PHP file comparison function to detect malicious files and restore legitimate ones.

**Module 3:** Quick scan module; uses keywords and characters to detect malicious files and restore legitimate files.

**Module 3.1:** List of keywords maintained in a file used to feed quick scan detection.

**Module 4:** Log Module; to record all the full and quick scan incidents.

**Module 5:** Email module; PHP email module to send the administrator quarantined and log files.

## **4.1 Ethical Disclosure**

We prioritise ethics and transparency throughout every aspect of our work. All documents were legally obtained and accessed through our university account during the literature review process. Our innovative artefact is developed using different built-in functions and used from the official WordPress development website, and the mailing function was sourced from GitHub and properly referenced. It does not store email addresses, instead utilising built-in functions to identify them. The malware data set was obtained from GitHub, which allows for open use. The testing domain and server are both owned by the author and not publicly disclosed. No harm was inflicted upon any organisations or individuals during the development and testing process. Our artefact is designed to not interfere with user data when deployed in the public sector.

## **4.2 Testing Environment and Dataset**

Due to frequent data transfer while downloading the WordPress and mailing logs, the plugin will not function properly in a localhost environment. Therefore, the domain <https://devmaltest.prbhat.com>, which belongs to the author, is utilised for testing purposes. The plugin requires at least 2 gigabytes of RAM, two gigahertz CPU, a 10 MB input/output threshold, and unlimited bandwidth. For seamless operation, the website should run with PHP 7.4 or later with WordPress 5.2. This standard low-configuration setup was chosen to showcase the plugin's optimised performance and ability to work in low-resource environments.

## **4.3 Classification Of Malware Used for Testing**

The latest malware package, which includes two packages from the GitHub repositories of Pejic and Petrak, was tested against the developed artefact. These samples have a commit history till 07 December 2022 and have impacted different attacks since 2016. Tools such as Malware Scanners and Simple Virus Scanners, commonly used in cPanel, rely on these collections of samples for detection. Details about two sets of malware collection are classified by uploading every file to the Virustotal and are mentioned under its malicious intention criteria in Table 03.

Sl. No	Webshell	Sl. No	Trojan / Trojan Downloader	Sl. No	Session or Backdoor Other than shell	Sl. No	Generic.PHP.malware
1	30.12.2020 - admin-post.php	28	23.12.2020 - wp-comments-post.php	55	16.12.2020 - kindex.php	79	26.10.2020 - ALL by APT29
2	10.12.2020 - Ccili.php	29	03.12.2020 - adsg.php	56	04.03.2020 - comment-abx.php	80	21.10.2020 - function.php
3	23.10.2020 - sy.php	30	25.11.2020 - wp-links-opml.php	57	10.05.2021 - ALL	81	06.02.2020 - index.php
4	22.10.2020 - bdhrlGW3ARq.php	31	19.11.2020 - ALL - index.php	58	26.04.2021 - new-index.php	82	10.06.2021 - wp-22.php
5	07.08.2020 - cm690ah4.php	32	27.10.2020 - index2.php	59	25.03.2021 - wp-alfa.php	83	21.03.2021 - ff.php
6	02.05.2020 - wp-demo.php	33	04.05.2020 - gistfile1.txt	60	18.02.2021 - baindex.php	84	01.02.2021 - apikey.php
7	04.04.2020 - orvshell_v2.php	34	03.02.2020 - linklove.php	61	08.02.2021 - windex.php	85	18.01.2021 - wp-flasher.php
8	04.03.2020 - spinner-gwb.png	35	30.09.2021 - lhwborcdhm.php	62	02.02.2021 - index.php	86	14.01.2022 - admin-post.php
9	01.12.2020 - api.neighbor.php	36	18.05.2021 - all - dd.php	63	12.01.2021 - admin.php	87	21.07.2022 - wp-config.php
10	16.09.2021- public_html(PHP)	37	30.04.2021 - see.php	64	11.01.2021 - page.php	<b>Sl. No</b>	<b>.htaccess or File Modifier</b>
11	15.09.2021 - bebas.php	38	11.02.2021 - wp-blogs.php	65	23.03.2022 - header.php	88	22.12.2020
12	29.06.2021 - 35be5.php	39	30.01.2021 - wwcgzjpcd.php	66	11.03.2022 - wp-blockup.php	89	08.10.2020 - ALL
13	14.03.2021 - s_e.php	40	26.01.2021 - ini.php	67	03.03.2022 - wp-xplrpc.php	90	08.09.2020 - replace.php
14	31.01.2021 - 8a8spqev.php	41	07.01.2021 - iskvrxtjqx.php	68	09.02.2022 - index.php	91	26.01.2021 - htaccess
15	06.01.2021 - index.php	42	05.01.2021 - iptajmthj.php	<b>Sl. No</b>	<b>DDos or Redirection</b>	92	10.08.2022 - 2index.php
16	19.09.2022 - index.php (ALL)	43	04.01.2021 - index.php	70	11.12.2020 - ALL	93	11.05.2022 - index.php
17	05.05.2022 - wp-config.php	44	07.12.2022 - POPA.php	71	04.12.2020 - ALL	94	29.04.2022 - ALL
18	04.05.2022 - class -wp-page.php	45	28.11.2022 - index.php	72	08.03.2020 - htaccess.php	95	15.04.2022 - .htaccess
19	26.04.2022 - Chitoge.php (ALL)	46	25.03.2022 - indee.xp	73	27.04.2021 - krz.htm	96	09.03.2022 - worker.php
20	18.04.2022 - index.php	47	21.02.2022 - qhrxh.php	74	01.06.2021 - index.php	97	28.03.2022 - wwdv.php
21	10.02.2022 - wp-settings.php	<b>Sl. No</b>	<b>Phishing or MalAds</b>	75	06.12.2022 - index.php	<b>Sl. No</b>	<b>Credential Exporter</b>
22	27.01.2022 - fck.php	48	04.02.2022 - All	76	10.03.2022 - ab19cleaf.php	98	20.11.2020 - index.php
<b>Sl. No</b>	<b>Multiple Signatures - Known</b>	49	08.10.2021 - All	77	11.02.2022 - xbnjfoobl.php	100	04.04.2020 - leafmailer.php
23	02.06.2020 - index.php	50	08.10.2021 - to.php	78	08.02.2022 - template-loader.php	101	07.02.2022 - index.php
24	15.03.2021 - ALL	51	10.03.2021 - index.php (Cred export)		-		-
25	12.03.2021 - ALL	52	13.01.2021 - mplugin.php		-		-
26	28.01.2021 - index.php	53	07.09.2022 - wp-lazyload-module.php		-		-
27	19.04.2022 - wp-crypto.php	54	04.02.2022 - index.php		-		-

 - Manually classified by decoding and research.

**Table 03. Classified malware based on its use and named based on date as mentioned in (Pejic, 2022).**

The first collection from Petrak contains forty thousand samples of different kinds of obfuscated JavaScript malware (Petrak, 2022) and is one of the best packages containing malware from 2015 to 2019. The second collection is from Pejic, with over a hundred malware samples covering the most attacks since 2019 (Pejic, 2022c). Both impacted websites and web applications in different files using file upload or injection vulnerabilities. Many of these still exist and are considered to impact badly either by stealing data, mining crypto from the browser, downloading ingress tools from the background, redirecting users to malicious sites, or scripts to compromise clients for Distributed Denial of Service (DDoS) over websites and servers; all these classifications cover most of the malware types being used in a real-time environment for the attack.

The Table of Scan along the report has the following attributes and represents them as mentioned below.

- Malware Tested - The malware used for testing.
  - Format of malware Date\_Detected - malware name with extension.
- Type - Type of attack method used for simulation.
  - Manual: Uploaded malware to a specific location as it attacks.
  - Injection Script - Appended malware to existing files using inject script to target specific files.
- Full Scan / Quick Scan - The type of scan performed, mentioning the relevant result.
  - Successful or unsuccessful in scan completion.
  - N/A signifies the scan is out of scope for the particular malware type.
- Time Taken - Time taken to finish the scan in seconds units.
- Comments - Review the result of the artefact.

These entities and attributes are carried out throughout the report.

Tested Malware	Type	Full Scan	Time Taken	Quick Scan	Time Taken	Comments
10.12.2020 - Coli.php	Manual	Successful	22.66177297	N/A	-	Full Scan completely cleaned website
19.04.2022 - wp-crypto.php	Manual	Successful	6.141340971	N/A	-	Full Scan completely cleaned website
22.12.2020 - 780 Malware files	Manual	Successful	8.517266035	N/A	-	Full Scan completely cleaned website
04.04.2020 - leafmailer.php	Manual	Successful	27.01757097	N/A	-	Full Scan completely cleaned website
04.02.2022 - All	Manual	Successful	6.207207918	N/A	-	Full Scan completely cleaned website
08.10.2021 - All	Manual	Successful	22.695158	N/A	-	Full Scan completely cleaned website
09.03.2022 - worker.php	Manual	Successful	6.079495192	N/A	-	Full Scan completely cleaned website
28.01.2021 - index.php	Inject Script	N/A	-	Successful	2.923151016	Qucik scan completely cleaned site
18.05.2021 - all - dd.php	Manual	Successful	6.263489008	N/A	-	Full Scan completely cleaned website
26.10.2020 - ALL by APT29	Manual	Successful	27.26613307	N/A	-	Full Scan completely cleaned website

**Table 04. Selected Malware that created critical issues in WordPress history.**

Table 04 consists of the top and highly volatile malware that caused issues on different websites worldwide. Mentioning the short description of each goes as follows.

**10.12.2020 - Coli.PHP** - “The malware itself is a PHP backdoor dropper that creates a custom function that uses curl and file\_put\_contents to download malware from a third party URL which the attacker provides in an HTTP request (GET or POST) to the infected website” (Luke, 2020). A tool used to download additional malware to gain control of the website and server used.

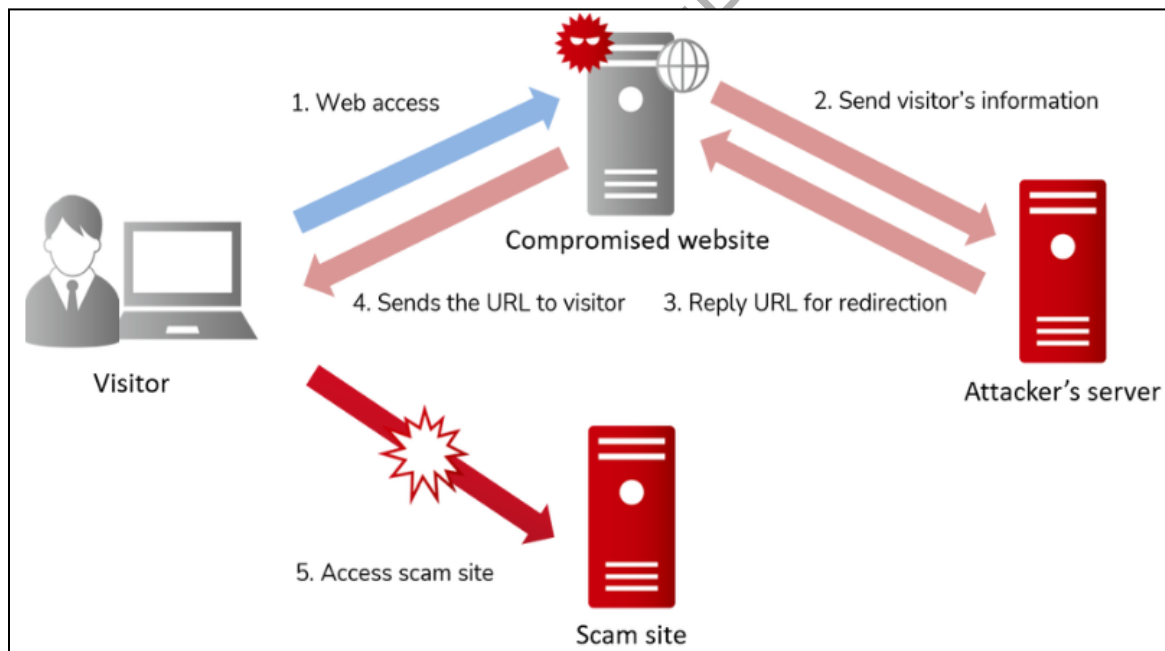
**19.04.2022 wp-crypto.PHP** - From the TrendMicro report, this malware behaviour is defined as: “This backdoor arrives on a system as a file dropped by other malware or as a file downloaded unknowingly by users when visiting malicious sites. It executes commands from a remote

malicious user, effectively compromising the affected system”(‘Backdoor.PHP.WEBSHELL.SBJKXRY - Threat Encyclopedia - Trend Micro BE’, 2022). The backdoor for the website and server is able to compromise the complete system.

**04.04.2020 - leafmailer.PHP-** Sucuri lab reports suggest, “Mailers is a category of scripts that hackers install on compromised servers to send out spam and anonymous emails” (‘PHP.hacktool’, 2019). A mailing malware used services as a spamming bot that indirectly caused blacklisting of the originating domain.

**26.10.2020-** ” *APT29 has been observed crafting targeted spear phishing campaigns leveraging web links to a malicious dropper; once executed, the code delivers RemoteAccess Tools (RATs) and evades detection using a range of techniques*” (Investigation, 2016). Altogether, this was one of the malware used by APT29 to achieve specific malicious goals.

An example of typical malware action that compromises a website and server, exploiting users and the target website, is illustrated in Figure 01.



**Figure 01: Idea of adversary acts on website and server. (Kino, 2021).**

The result section discusses other malware and its effects as it caused considerable changes to the website.

## 4.4 Modules Developed and Their Functions

### 4.4.1 Module 1: Activation Module; download WordPress from GitHub and create necessary files and folders for operation.

Activation of the plugin is a point where the plugin gets calibrated according to WordPress configuration. Here, folders such as backup, quarantine, and WordPress are created. The backup directory serves as a designated storage location for custom files related to a specific website, with a focus on the wp-config.php file. The implementation of this concept helps to efficiently organise and scan these critical files, thereby promoting the smooth functioning of the full scan. The 'wp-config.php' file is automatically copied to the backup folder and emailed to the administrator each time after activation. The process of copying and mailing is of paramount importance as it preserves the website-specific configuration. The importance of this procedure is outlined in Module 5, which has the entire structure of the email module.

After that, the quarantine folder serves as storage for any infected files, misconfigured or uploaded with malicious intent. The WordPress folder is employed to maintain an unzipped version of WordPress. This allows for the retrieval of a genuine copy of the WordPress core files for each iteration of the full scan. The process of re-downloading a new copy of WordPress for each full scan serves a dual purpose. Firstly, it ensures that any potentially infected files are replaced during the scan. This helps to maintain the integrity of the artefact by providing an additional layer of protection. Secondly, the periodic downloading of a fresh copy of the necessary files helps to confirm new files are used for every scan.

```
public function download_wordpress() {  
    # wordpress version  
    global $wp_version;  
    $file_name = 'wordpress-' . $wp_version;  
    $url = "https://wordpress.org/{$file_name}.zip";  
    $file_path = WP_KEEP_UP_RESOURCE . $file_name . '.zip';  
    $file = fopen($file_path, "w");
```

Figure 02. Code snippet for downloading a new copy of the installed version.

#### 4.4.2 Module 2: Full Scan Module uses the PHP file comparison function to detect malicious files and restore legitimate files.

A simple yet effective method to scan all the files in the WordPress file system, and every file in the website will be compared except files in the wp-content directory. Using the *if* condition, as in Figure 03, every existing running file is compared against the downloaded files from the plugin WordPress directory. In the event that a discrepancy in characters is detected between the running file and the downloaded file, those files will be removed and replaced with a legitimate downloaded file. This approach avoids the downtime typically caused by other plugins, which often require the replacement of the entire backup or extensive manual verification through every file.

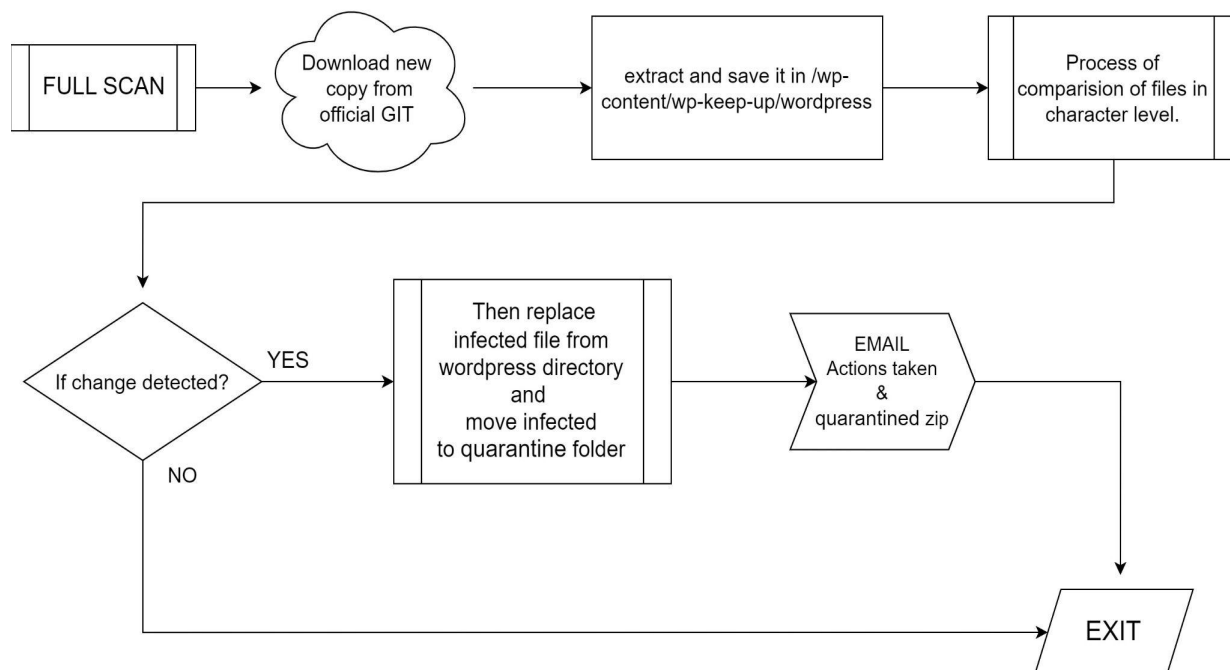
```
$file_content = file_get_contents($file); // check for file content
$source_content = file_get_contents($source); // with legit

if($file_content != $source_content) { // comparision of files
```

**Figure 03. Code snippet for comparing existing and new copies of files.**

Upon completing the full scan, any infected files will be moved to the quarantine directory and compressed. The quarantine folder will then be emailed to the administrator using the PHP mail function. ('wp\_mail() | Function', n.d.). This step will help the cyber forensic team to investigate the attack, understand the malware type, and block these attacks from occurring again.





**Flowchart 01: Flowchart of the process flow of the full scan module.**

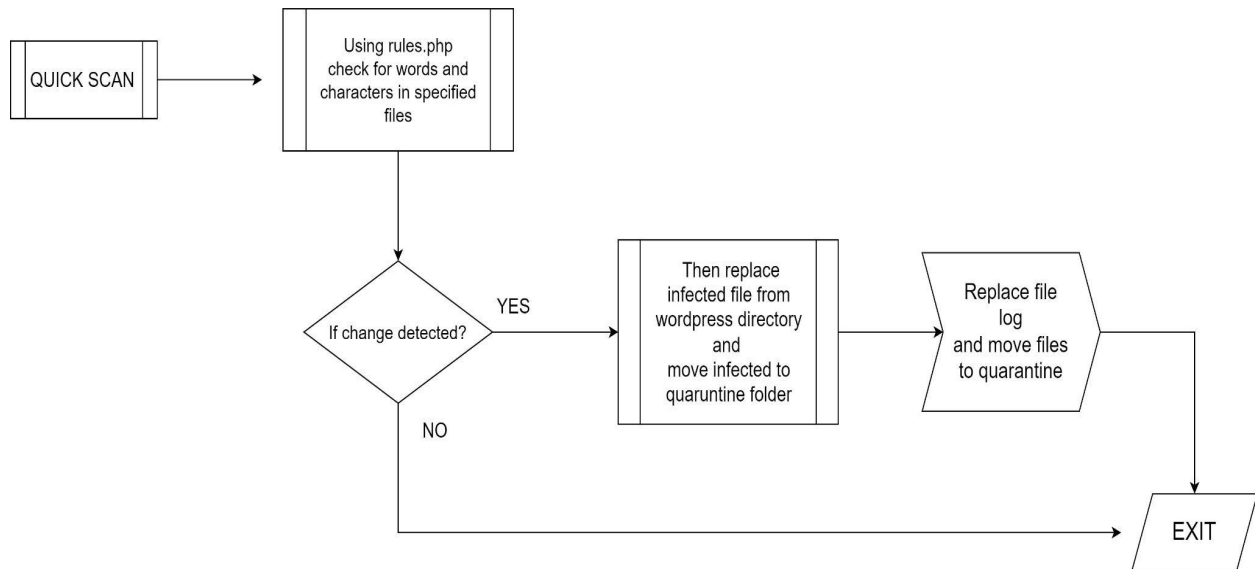
Using a simple WordPress cron job, we can automate the full scan process. The WordPress cron job is a scheduled task that automatically triggers at predetermined intervals within the WordPress environment. These tasks include checking for updates, publishing scheduled posts, and optimising the database. They can be set to occur at specific times or regular intervals. The full scan module can be accessed through the URL: [https://\[domain\]?run=wp-keep-up&type=full-scan](https://[domain]?run=wp-keep-up&type=full-scan), and can be set to run at specific intervals as a cron job to meet the needs of the administrator. Overall, an explanation of the full scan process involves the detection, replacement, recovery and restoration, and logging of all actions ends here.

#### **4.4.3 Module 3: Quick Scan Module; use keywords and characters frequency set to detect malicious files and restore legitimate files.**

The concept of the quick scan here is to check the specific files against the list of words and characters often found in malicious scripts. If there is a match or some characters are used excessively than originally required, they will be moved to quarantine and replaced with begin file from the downloaded directory.

The quick scan only includes a specific set of files, such as index.php, wp-links-opml.php, wp-trackback.php, xmlrpc.php, wp-settings.php, wp-mail.php and wp-cron.php. It is worth noting

that, according to data from a recent attack on November 15, 2022 (Rees, 2022), all of the above-mentioned files were affected. This attack report motivated the inclusion of these files in the quick scan, as they are known to be frequently and regularly targeted in many other attacks.



**Flowchart 02. Flowchart of the process flow of the quick scan module.**

The quick scan module can be accessed through the URL: [https://\[domain\\_name\].com?run=wp-keep-up&type=quick-scan](https://[domain_name].com?run=wp-keep-up&type=quick-scan) and can be set to run at specific intervals as a cron job to meet the needs of the administrator.

#### **4.4.4 Module 3.1: List of keywords maintained in a file that feeds quick scan detection (rules.php).**

The purpose of this module is to compile a list of keywords and characters into a specific file which can then be used to feed the Quick scan module. This list is developed based on an analysis of keywords from the literature review and additional research on the latest WordPress malware reports from (Patchstack, 2022) and Sucuri. The characters and words are checked individually by placing them into an array. A code snippet which shows the structure of keywords and characters used under the array to feed a quick scan is shown in Figure 04 of the rules.php file.

```

array(
  'files'      => array('index.php'),
  'characters' => array( '-' => 5, '(' => 2, ')' => 2, '@' => 3, '*' => 2, '<' => 2, '*' => 17, '?' => 2
  'words'      => array('eval' > 1, 'setTimeout' => 1, 'document.addEventListener' => 1, 'classid' => 1,
  'file_type'  => 'core'
),

```

**Figure 04.** The code snippet of the rules.php file; shows words and characters used for matching.

#### 4.4.4 a) Algorithm of the quick scan process

**Step 1:** Start.

**Step 2:** Select files from the array that needs to be checked against rules.php.

**Step 2.1:** From rules.php, initialise the array with words.

# Set malicious keywords that must not be present in the selected file.

**Step 2.2:** From rules.php, initialise the array with characters.

# Set of malicious characters that must not be present in the selected file.

**Step 3:** If Step 2.1 or 2.2 is true, move the infected file to quarantine.

**Step 4:** Replace the infected file with a clean one from the downloaded WordPress directory.

After step 4, the logging function is called again to log activities similar to a full scan.

#### 4.4.5 Module 4: Log Module; To record all full and quick scan incidents.

Documentation of the actions taken is part of understanding and transferring knowledge to users and administrators. The logging module brings the quantitative report of all actions taken in a full and quick scan. Here, events defined as 'Date Time, Scan Type, Corrupted file Char Count, Original file Char Count, File Name, File Path, File Permission, Action Taken and File Relation' are recorded for every file. The code snippet for logging is shown in Figure 05.

The logging module creates a log.csv file with header information during plugin activation. This file serves as a record for website administrators and cybersecurity analysts to understand the incident and conduct a thorough investigation. Each time a full scan or quick scan is performed, the actions taken are appended to the log.csv file.

```

$csv_header = array('Date', 'Time', 'Scan Type', 'Suspicious Char Count', 'Original Char Count', 'File Name', 'Path', 'Permission',
# Log file
$log_file = WP_KEEP_UP_RESOURCE . 'log.csv';
if(!file_exists( $log_file )) {
    touch($log_file);
    $file = fopen($log_file, 'a');
    fputcsv($file, $csv_header);
    fclose($file);

```

**Figure 05.** Code snippet used to generate log.csv file.

#### 4.4.6 Module 5: Email Module; PHP mail module to send the administrator quarantined files and log files.

A PHP function called 'wp\_mail()' is used to send emails to the administrator during the installation process. This method helps to keep files even safe as the administrator will have a backup copy in email. Saving the wp-config.php file is essential because it contains website-specific information. It is crucial to save the wp-config.php file because it contains configuration specific to a particular website. If the website is attacked and the wp-config.php file becomes infected, even if it is in the backup folder, the mailed copy can be used to restore the old settings instead of relying on a traditional complete backup restoration.

Furthermore, this mailing function is invoked only for the full scan process, sending a quarantine compressed zip file with the latest log to the website administrator. The mailing process may cause a short delay due to server traffic and resource availability. To speed up the quick scan process, the mailing action is not invoked. However, the quick scan logs are still saved and appended to the log.csv file, which can be retrieved from the 'Download Log' button or the following email of the full scan.

```
// mail to , subject etc..
$to = get_bloginfo('admin_email');
$subject = 'WP Keep Up - Activated';
$date = date('Y-m-d H:i:s');
$body = "<h1>WP Keep Up - Activated</h1><p>Activation Date: $date</p>";
$headers = array('Content-Type: text/html; charset=UTF-8');
if(wp_mail( $to, $subject, $body, $headers, $config_text )) {
```

Figure 06. wp-mail() - a PHP mailing code snippet from the WordPress development repository.

Continuing to the next section, the plugin's performance is analysed using a tool called p3-plugin-profiler, which was developed by the popular hosting and web services company Godaddy (wpmudev/p3-profiler, 2022).

### 4.5 Artefact Performance Analysis And Comparison

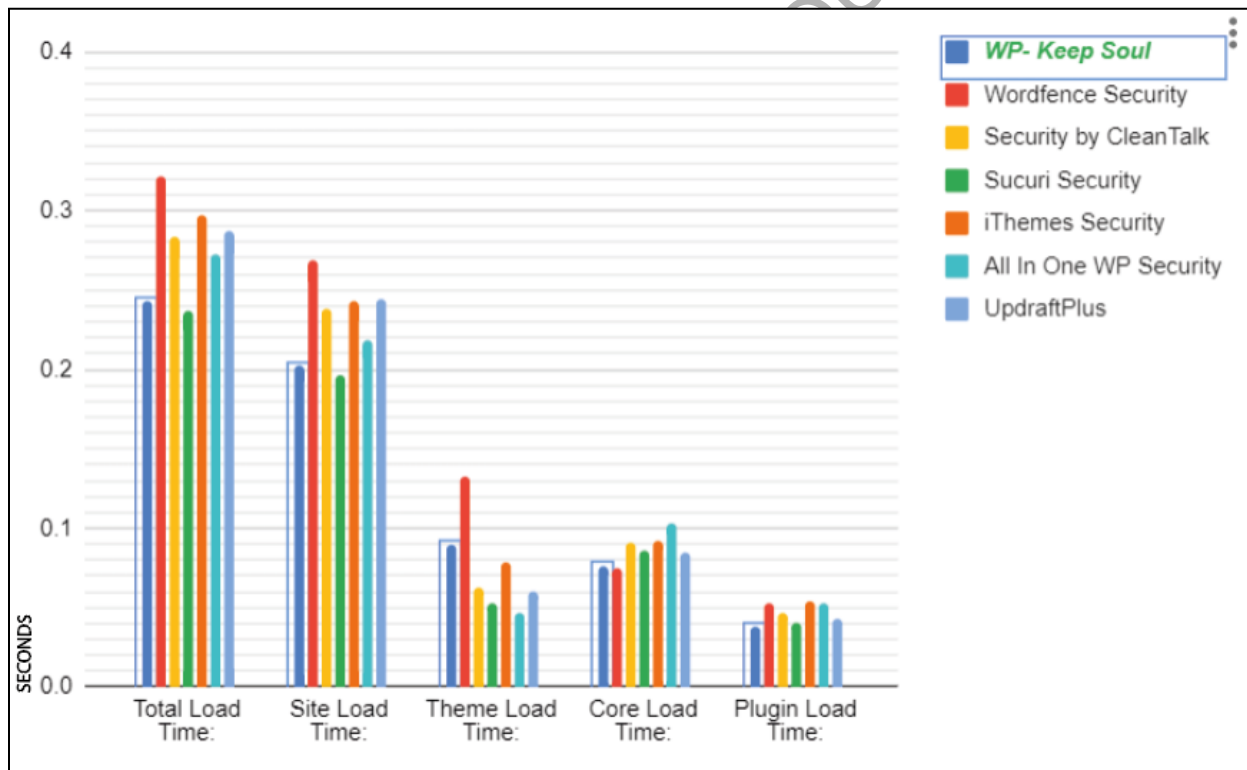
It is essential to verify the key characteristics of the artefact prior to utilising its capabilities. Performance analysis is vital before use to ensure that the artefact is compatible in a running environment. We need to consider how the installation of this tool may affect the core infrastructure, including its impact on the loading and refresh times of the entire infrastructure. Figure 8 presents a comprehensive analysis of plugin performance, including runtime by plugin and the impact on the theme, plugins, and core files. It also includes comparing other plugins

and the website's loading time before and after the plugin's installation. All the points discussed are in the test results section below, and the functionality test of detection and other mechanisms is discussed in the results and evaluation section.

The plugin used for comparison is based on popularity, and result relevance in WordPress search are listed below:

- Wordfence Security
- Security & Malware scan by CleanTalk
- iThemes Security
- WPScan
- Sucuri
- All-In-One Security (AIOS)

(the profiler tool did not recognise some other popular plugins; hence, it is excluded from analysis)



**Figure 07.** The chart represents the time different plugins take for different metrics.

In Figure 07, WP-KeepSoul metrics are highlighted, and the characteristics of WP-KeepSoul vs other plugins are explained below.

**Total Loading Time:** Least among others except for Sucuri Security, with a few milliseconds more.

**Website Loading Time:** The metrics are similar to the total loading time.

**Theme Loading Time:** Wp-keepsoul will impact theme loading time due to the PHP mailing function.

**Core Loading Time:** The difference is ignorable but has negligible impact.

**Plugin Loading Time:** Here, wp-keepsoul will use the least time to load than all others.

## 4.6 Building Attack Simulation

The malware collection used in this study was obtained from the GitHub repository (Petra, 2022), which includes a range of malware samples dating from 2015 to 2022. Many of the malware samples in the collection were detected by existing anti-malware services. Despite efforts to combat malware, a significant number of malicious software programs remain active and continue to cause problems for website administrators. The goal is to deal with real and catastrophic trojans, which leads to total compromise of the website and makes it unrecoverable. All the malware were checked against virus total and classified based on relevant results; some of the types of malware used for testing are:

- Webshell
- Phishing
- Trojan
- Trojan-Downloader
- File Modifier(.htaccess) Multiple Malicious Intention
- Backdoor
- DDoS
- Redirection
- Generic.PHP.malware
- Credential Exporter

We have already discussed the intention and intensity of malware in the literature review and the methodology section.

## 4.7 Attack Simulation

In a real-time environment, the attackers exploit the vulnerabilities in various plugins and upload or inject malware. In this testing case, an injection script was developed to upload the malicious zip file and simulate the attack.

### 4.7.1 First Method: Injection Script Working

Injection code is developed to aid in the testing phase by automating the process of appending malicious code snippets, rather than manually performing attacks. This helped to inject 3000 malware samples during the testing phase in a short time. The script simulates the injection process that could occur through attacks such as reverse shell, exploiting a file upload vulnerability and plugin vulnerability.

To use the injection code, a tester should first place the script and create a lib folder in the root directory of the target website. They should then add malicious files to the lib folder and run the inject.PHP script using the domain-specific URL, such as `www.example.com/inject.php`. It is important to note that this kind of access can be obtained through an exploit by an attacker or by a legitimate administrator who has access to their own website and server file system.

The working of script is simple; the PHP script randomly picks a file from the 'lib' folder, then injects it into the target file and exits successfully with a blank page. Conversely, if the target file is not present in the WordPress directory, it returns 'Target File Not Found' as shown in Figure 08 below. Also, this code is developed only to inject for one particular file selected randomly.

```
$file_absolute_path = __DIR__ . '/' . $file;
if(file_exists($file_absolute_path)) {
    $malicious_code = file_get_contents( getRadomFile( $lib_path ) );
    $original_code = file_get_contents( $file_absolute_path );
    file_put_contents( $file_absolute_path, $malicious_code . $original_code );
} else {
    echo '<p><b>Target File Not Found </b>' . $file . '</p>';
}
```

Figure 08. Code snippet from inject.PHP script used.

The complete code flow as an algorithm:

- Step 1: Define lib path.
- Step 2: Define target files for injection.
- Step 3: Get the malicious file contents and append them to the target file before the

actual file content // the usual malware working method is simulated.

Step 4: If the target file is not present in the destination,  
then return the target file not found.

Step 5: The blank page is loaded if the attack is successful.

Step 6: exit.

#### **4.7.2 Second Method: Manually place specific malware**

Malware which uses more than one file and does not get appended but works as an ideal file to disrupt the regular usage of the website or act as reverse shell, PHP backdoor, or credential exporter are manually placed by uploading to a particular directory of the website, For example, 20.10.2020 - APT29.

PRANEETHRAJ WP KEEPSOUL Version 1.0



# CHAPTER 5. TEST RESULTS

To effectively test the artefact, it is essential to understand the boundaries and scope of the testing. The following points outline the scope and test criteria:

- Access to the WordPress dashboard is necessary for the successful execution of the scanning process.
- Any malware that causes lockout of the admin dashboard is excluded from our project scope.

## 5.1 Malware Test Results

The test successfully achieved all the aims and objectives by effectively removing malware, quarantining it, replacing it with legitimate files, and logging all actions taken. Also, it was able to detect unwanted files and delete them. The minimum time required to complete detection and mitigation in the full scan is recorded to be 6.0140249729156 seconds, and the maximum time consumed is 44.44684792 seconds. The test was conducted by randomly picking 3000 malware from the (Petra, 2022) collection and adding it to random files WordPress to test the artefact for the detection and prevention module of a quick and full scan. Conversely, it was not recorded for analysis as the classification was too challenging to finish in the timeframe. Despite this, 91 out of 101 classified malware samples were used for testing, as the others were out of scope as they targeted the wp-content directory. The analysis of the results will be discussed in the subsequent evaluation section.

### 5.1.1 Result Comparison Criteria

Compared with Wordfence, the average scan time that the company says is around 1 to 10 minutes based on the website size ('Scan', n.d.). Likewise, Malcare also states, "*Guaranteed WordPress Malware Removal in 5 minutes*" ('WordPress Malware Removal Service - No Wait Time, 1 Click Malware Cleaner', n.d.). Most of the solutions available require an active paid subscription to use. As a result of the limited scope of malware samples utilised, the detection rates and metrics, such as the time taken for the process, cannot be comprehensively evaluated. Furthermore, an examination of the official websites and documentation of various solution providers, including i-Themes, Snap Creek, Clean Talk, WP Vivid, Backup Guard Security, and Bold Grid, reveals that they do not explicitly provide information regarding the

duration required for the completion of a scan and proper mitigation; instead, they only mention the resource requirements for their solutions.

### 5.1.2 Effects of Malware Before Scan

During the testing phase, some malware made visible changes to the website. Since these were discovered during the testing phase, it is relevant to mention some of the results in this section.

The figures below (10, 11, 12, and 13) show malware's effects on the site after execution.



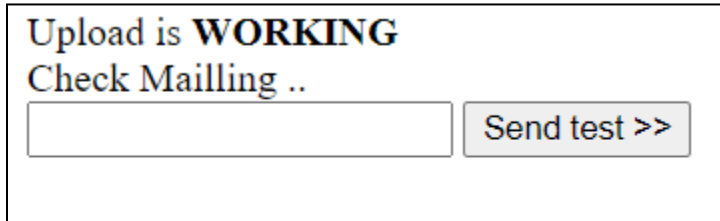
Figure 09. 28.01.2021 - index.php malware effects on the testing website.

As shown in Figure 09, the exploit caused the front end of the website to crash, and all previous content was lost. Although, the dashboard was still accessible while the scan was performed and the website was cleaned.



Figure 10. 09.03.2022 worker.php malware effects on the testing website.

worker.php malware creates a form that can be executed on the server's operating system, which indirectly can take over without harming the website; the website owner will not know where this malware resides. Advisory can access these just by calling the malicious PHP file.



Upload is **WORKING**  
Check Mailling ..

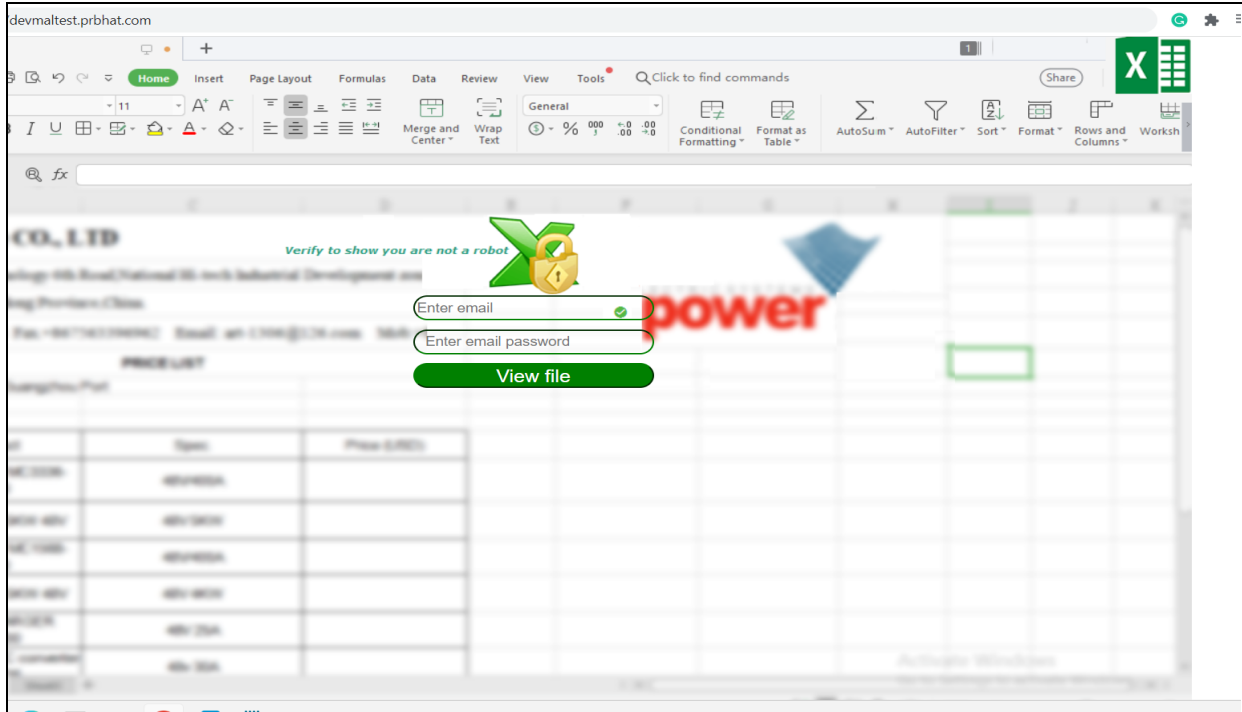
Figure 11: 08.10.2021 malware effects on the testing website.

Another malicious PHP file that creates the above page overrides the existing wp-content folder and creates content as website content.



Figure 12. 18.05.2021 malware effects on the testing website.

Malware that denies users access to the page; only if there is access to the dashboard can the website be recovered by wp-keep-soul or as regular, the administrators must restore to the previous state of the website.



**Figure 13. 04.02.2022 malware effects on the testing website.**

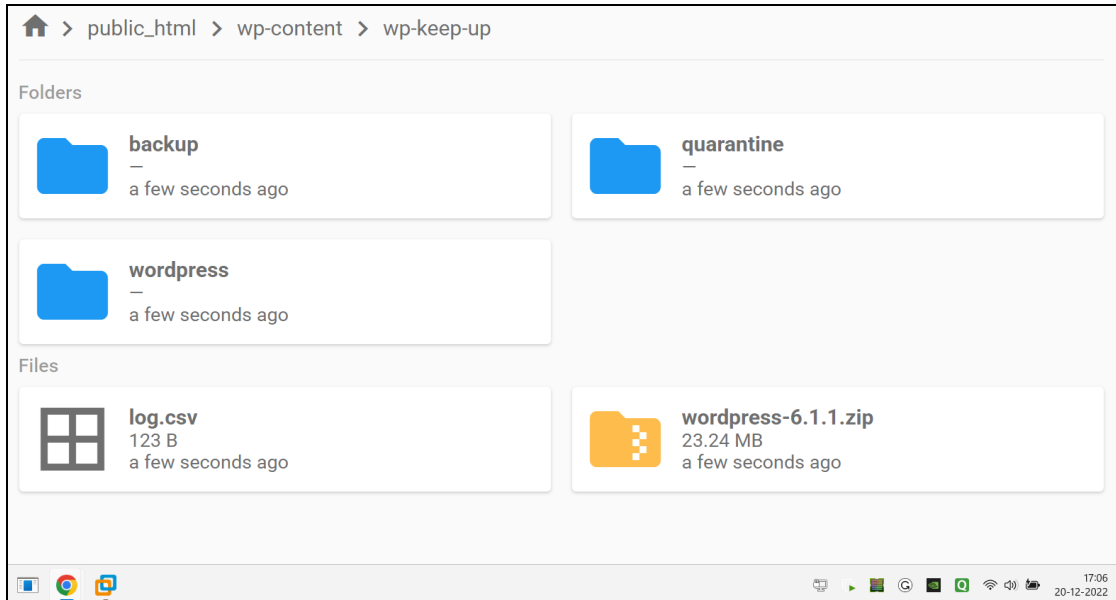
A phishing malware creates a whole new page similar to an excel sheet in the background but blurs the image with a popup that asks for email and password to end, but this sends an email to the attacker's mailbox with input credentials.

## 5.2 Module Test Results

The section is again divided according to the module, and their results are discussed with relevant result tables and screenshots representing actions and effects of actions taken.

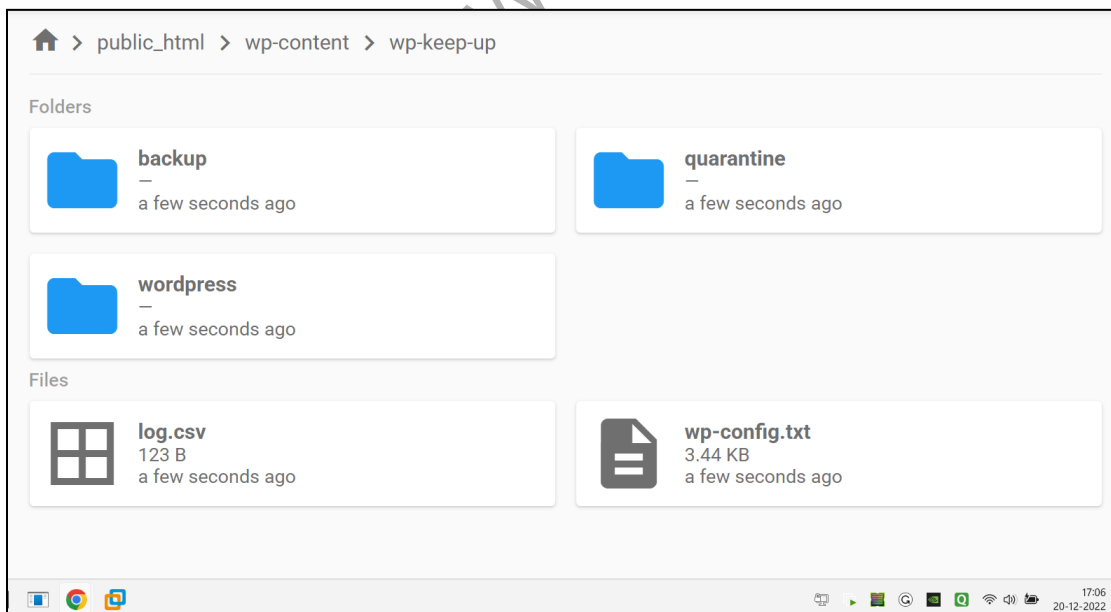
### 5.2.1 Module 1: Activation Module

The algorithm is already mentioned in the methodology, the functioning of activation is initiated first, and it creates the required files and folders to function normally. Figure 14 and 15 describes the folders created during activation.



**Figure 14. Creation of directories and downloading WordPress.**

As described in the methodology section, directories such as backup, quarantine, and WordPress are immediately generated when the activate button is pressed, as shown in Figure 14. The main thing to observe is a WordPress-6.1.1.zip file downloaded from git and then extracted and placed inside the WordPress directory, which supports all the detection.



**Figure 15. Creation of directories and downloading WordPress.**

A few seconds after the activation and downloading process, the zip will be deleted, and the wp-config.txt file appears, which is then used to mail the administrator; a copy will be automatically stored in the backup folder. Figure 15 shows a screenshot of the administrator's Gmail account after activating the plugin. The administrator receives a notification containing the date and time of activation and the wp-config.php file for reference.

## 5.2.2 Module 2: Full Scan Module

The full scan module's primary functionality is to scan all files, compare them and restore them if they fall into the algorithm of full scan mentioned in the methodology. The complete full scan result is mentioned in Appendix D. Results show that all the criteria of the full scan are met, and the methodology is satisfied by the proper output of a malware-free website. Some of the cases are listed in Table 05, which shows some examples.

Tested Malware	Type	Full Scan	Time Taken(Sec.)	Comments
01.12.2020 - api.neighbor.php   wp-load.php	Manual	Successful	44.44684792	Full Scan completely cleaned website
16.09.2021- public_html(PHP)	Manual	Successful	6.326412916	Full Scan completely cleaned website
15.09.2021 - bebas.php	Manual	Successful	6.27789402	Full Scan completely cleaned website
29.06.2021 - 35be5.php	Manual	Successful	6.020296812	Full Scan completely cleaned website
31.01.2021 - 8a8spqev.php	Manual	Successful	6.300168037	Full Scan completely cleaned website
19.09.2022 - index.php (ALL)	Manual	Successful	7.114274025	Full Scan completely cleaned website
04.05.2022 - class-wp-page.php	Manual	Successful	17.52194405	Full Scan completely cleaned website
10.02.2022 - wp-settings.php	Inject Script	Unsuccessful	-	Site Crashed
27.01.2022 - fck.php	Manual	Successful	6.294228077	Total Clean and Malware Free Result
18.02.2021 - baindex.php	Manual	Successful	6.128773928	Full Scan completely cleaned website
08.02.2021 - windex.php - all	Manual	Successful	6.461483955	Full Scan completely cleaned website
12.01.2021 - admin.php -wp-admin	Inject Script	Unsuccessful	-	Malware Overrided admin dashboard
11.01.2021 - page.php -all	Manual	Successful	6.153823137	Full Scan completely cleaned website
23.03.2022 - header.php	Manual	Successful	6.075851917	Full Scan completely cleaned website
11.03.2022 - wp-blockup.php	Manual	Unsuccessful	-	Malware Overrided admin dashboard
03.03.2022 - wp-xplrpc.php	Manual	Successful	6.064377069	Full Scan completely cleaned website

**Table 05. Full Scan results; Limited (complete list in Appendix D)**

Table 05 is filtered results from a list of 91 different scans based on the scan's malware effects and success criteria. Even though it detected almost every malware, there were three cases out of 3091 scans where the WordPress dashboard was hijacked by malware and was unable to take any actions; all three unsuccessful cases are discussed as follows:

**Case 1: 10.02.2022 wp-settings;** the malware caused a 500 error (internal server error) and crashed, it was resolved by manually deleting the infected wp-settings file and replacing it with a legitimate one.

**Case 2: 12.01.2021 admin.PHP**, The second case failed as malware infiltrated the wp-admin directory, compromising the dashboard. Unable to remove it manually, the only solution was to restore from a backup.

**Case 3: 11.03.2022 wp-blockup.PHP**, The malware blocked the dashboard, adding a script in other admin files, making it impossible to recover manually; the solution was restoring the entire website from a backup.

All cases, 1, 2, and 3, signify the excellent results of the module, and some limitations highlighted are in the evaluation and discussion section.

### **5.2.3 Module 3: Quick Scan Module**

The primary function of the quick scan is to examine files for specific characters and keywords to detect malware; The results were consistent and completely satisfied the requirement. The mitigations implemented by the quick scan were determined to be effective, resulting in a thoroughly cleaned website.

Based on the test data utilised, which consisted of 91 instances of malware, it was found that a significant proportion of these was dependent on the file "index.php". Such malware operates by being triggered indirectly upon loading the site. Later, a quick scan was performed and successfully mitigated the malware. Other than the index, files like wp-links-opml.PHP and wp-blogs.php are scanned and successfully mitigated. Table 06 represents the efficiency of a quick scan.

Tested Malware	Quick Scan	Time Taken(Sec.)	Comments
14.03.2021 - s_e.php   index.php	Successful	2.899857998	Qucik scan completely cleaned site
06.01.2021 - index.php	Successful	2.805902004	Qucik scan completely cleaned site
26.04.2022 - index.php (ALL)	Successful	2.798499823	Qucik scan completely cleaned site
18.04.2022 - index.php	Successful	2.817168951	Qucik scan completely cleaned site
02.06.2020 - index.php	Successful	3.02203083	Qucik scan completely cleaned site
28.01.2021 - index.php	Successful	2.923151016	Qucik scan completely cleaned site
25.11.2020 - wp-links-opml.php	Successful	3.035900116	Qucik scan completely cleaned site
11.02.2021 - wp-blogs.php	Successful	2.899831057	Qucik scan completely cleaned site
04.01.2021 - index.php	Successful	2.895252943	Qucik scan completely cleaned site
28.11.2022 - index.php	Successful	2.729642868	Qucik scan completely cleaned site
02.02.2021 - index.php	Successful	2.80268693	Qucik scan completely cleaned site
09.02.2022 - index.php	Successful	2.766499043	Qucik scan completely cleaned site
04.12.2020 - index.php	Successful	2.749493122	Qucik scan completely cleaned site
01.06.2021 - index.php	Successful	2.917783022	Qucik scan completely cleaned site
06.12.2022 - index.php	Successful	2.830790997	Qucik scan completely cleaned site
11.02.2022 - index.php	Successful	0.060942173	Qucik scan completely cleaned site
06.02.2020 - index.php	Successful	2.729393005	Qucik scan completely cleaned site
14.01.2021 - index.php	Successful	2.898641825	Qucik scan completely cleaned site
11.05.2022 - index.php	Successful	2.861639977	Qucik scan completely cleaned site
20.11.2020 - index.php	Successful	2.795728922	Qucik scan completely cleaned site
07.02.2022 - index.php	Successful	2.885725021	Qucik scan completely cleaned site
Average Time =		2.720312459	

Table 06. Quick Scan Results.

From all the above results, it is clear that Modules 2, 3 and 3.1 work according to the methodology as scans were successful.

#### 5.2.4 Module 4: Log Module

Another essential function records all the logs of actions. Figure 15 shows the log.csv file generated during the activation; this file is generated during the activation of the plugin, and data gets appended whenever the scan happens. Table 07 below represents four distinct types of entries in the log, which were selected from the malware tested in Modules 2 and 3.

Sl. No.	Date	Time	Scan Type	Suspicious Char Count	Original Char Count	File Name	Permission	Action Taken	File Type
1	12/19/2022	2:20:20	Full Scan	17803	No backup	orderhistory.php	755	Removed	
2	12/19/2022	1:33:06	Quick Scan	4364	405	index.php	755	Replaced	Core file
3	12/19/2022	1:55:36	Full Scan	41409	34350	wp-signup.php	644	Replaced from Source	
4	12/19/2022	0:40:07	Full Scan	3371	3527	wp-config.php	644	Replaced from backup	

Table 07. Entry of four different cases of logging.



The first entry in the log pertains to the full scan results of malware detection. It was found that a file located in the root directory was infected with malware, and the action taken was to remove it. This entailed deleting the file from the root directory, relocating it to the quarantine folder, and sending a notification email.

The following quick scan result shows that the suspicious char count field denotes the characters present in the infected file, and the original char count field says how much it is according to WordPress. Based on the condition, the action taken is to replace the file with the original file. As it is a quick scan and it is necessary to show the index.php is what type of file in the last column, as index.php is also present in many other directories of WordPress.

Sl.No 5 represents the result of another full scan, which is similar to the previous case, but here 'replaced from the source' is mentioned to help the log analyser. Nevertheless, from the action perspective, the file will be replaced from the WordPress directory of the plugin.

In the last row, the result of the full scan shows the action taken as replaced from the backup. First, the analyser needs to observe that the file wp-config.php is infected, and it is scanned from the backup folder.

The complete logs of detected malware are presented in Appendix G, and a snapshot from it is shown in Figure 16.

Sl No	Date	Time	Scan Typ	Suspicious Cha	File Name	Path	Permission	Action Taken
1	12/19/2022	0:02:17	Quick Scan	2145	index.php	/devmaltest.prbhat.com/public_html/index.php	755	replaced
2	12/19/2022	0:02:47	Full Scan	34	.maintenance	/devmaltest.prbhat.com/public_html/.maintenance	755	Removed
3	12/19/2022	0:02:47	Full Scan	3306	wpload.php	/devmaltest.prbhat.com/public_html/wpload.php	755	Removed
4	12/19/2022	0:02:47	Full Scan	1	hydra.pid	/devmaltest.prbhat.com/public_html/hydra.pid	755	Removed
5	12/19/2022	0:03:15	Full Scan	3306	wp-load.php	/devmaltest.prbhat.com/public_html/wp-load.php	755	Replaced from Source
6	12/19/2022	0:03:15	Full Scan	3098657	.bt	/devmaltest.prbhat.com/public_html/01.12.2020/.bt	644	Removed
7	12/19/2022	0:03:15	Full Scan	1742850	domains.txt	/devmaltest.prbhat.com/public_html/domains.txt	755	Removed

Figure 16: Snapshot of logs recorded. Appendix G has the complete list.

### 5.2.5 Module 5: Email Module

The purpose of the email function is to notify the administrator of the website about the installation of the plugin and also send a copy of the configuration file as a precaution. In the second phase, the email is to keep a record of the full scan, all the logs and quarantine files.

Figures 17 and 18 below demonstrate the various actions of the email function that are activated upon initiation. This function sends a message about activation, along with the wp-config.txt file, to the administrator. As evident from Figure 17, it is observed that the function is working as expected.

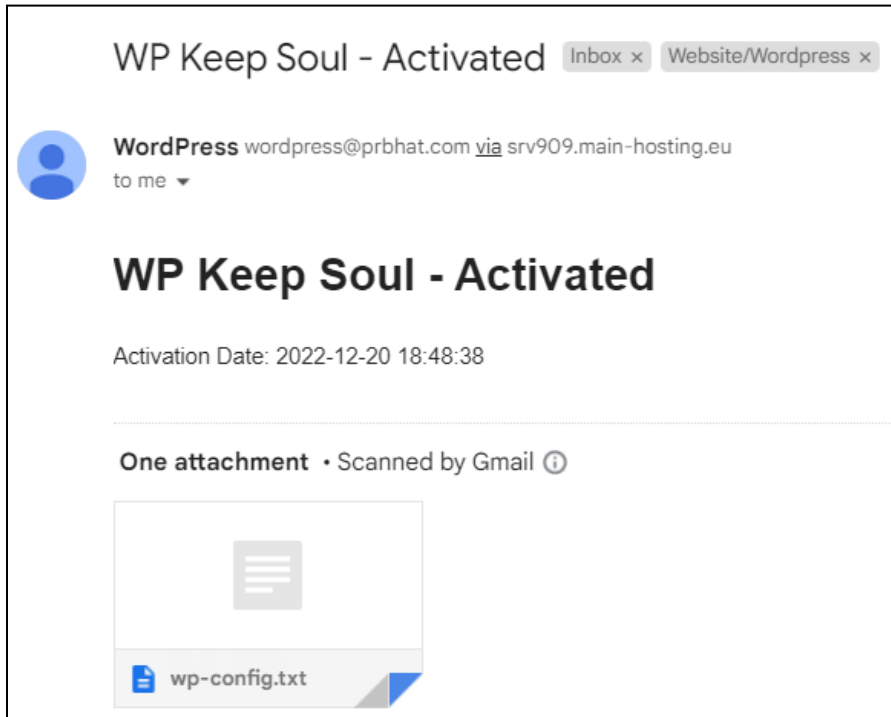


Figure 17. Email received after plugin activation.

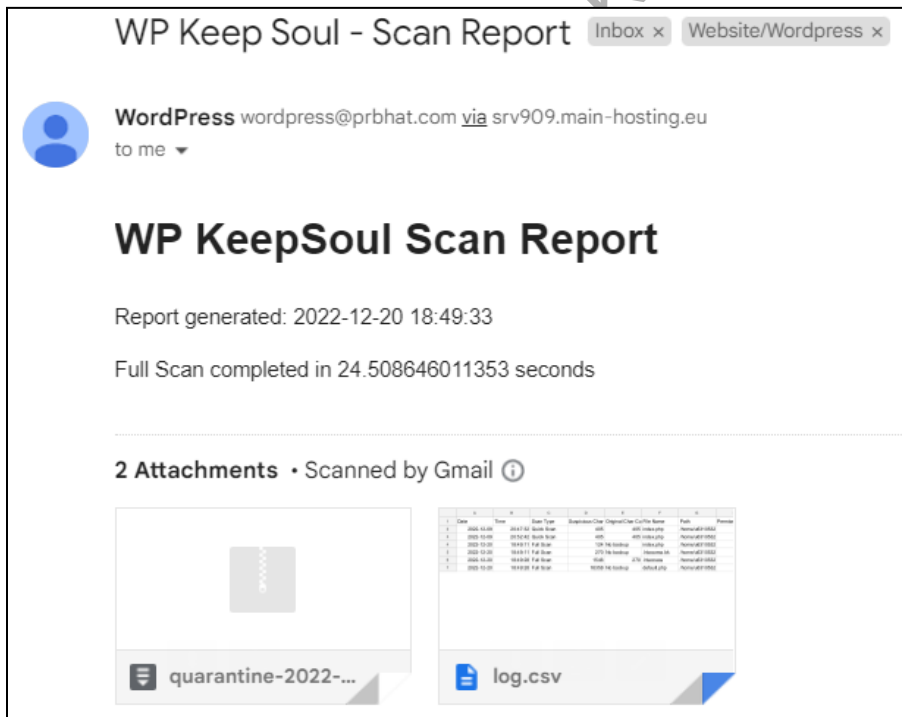


Figure 18: Email received after a full scan.

Figure 18 shows that scan reports mailed to the administrator contain a log.csv file, which is the scan log, and a zip file, which is the quarantine of infected files. This email function gets triggered after the completion of the full scan. The further upgradation that is required for the mailing function is discussed in challenges and future work.

### 5.1.7 Artefact in Action

The final development of all the modules led to the output of artefact WP-KeepSoul, which performed the actions required. The relevant snapshot and the explanation regarding each figure are discussed below.

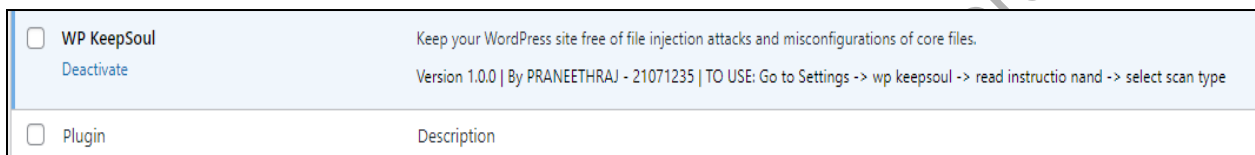


Figure 19. WP KeepSoul after activation on the dashboard page.

Figure 19 represents the snapshot from the plugin page of the WordPress dashboard; this is the first phase after the installation plugin in which the plugin can be activated and deactivated.

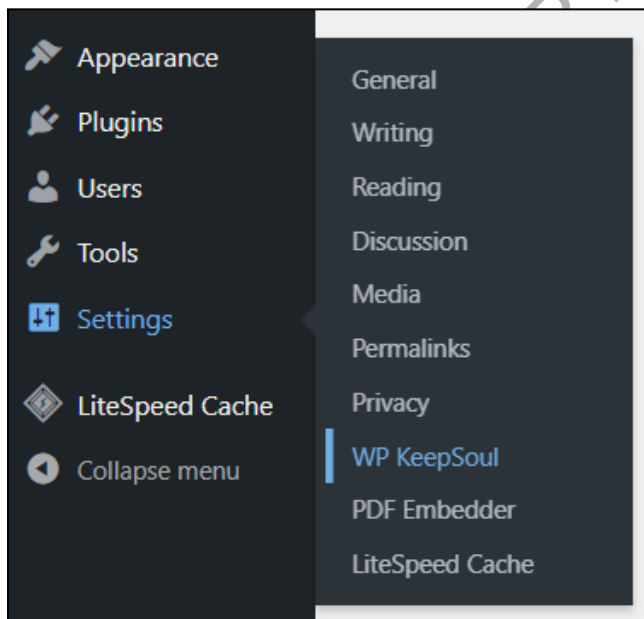


Figure 20. WordPress Dashboard Settings panel, which has plugin access.

Figure 20 represents the settings menu which contains wp-keepsoul. This menu appears in the side panel, giving access to the complete plugin.

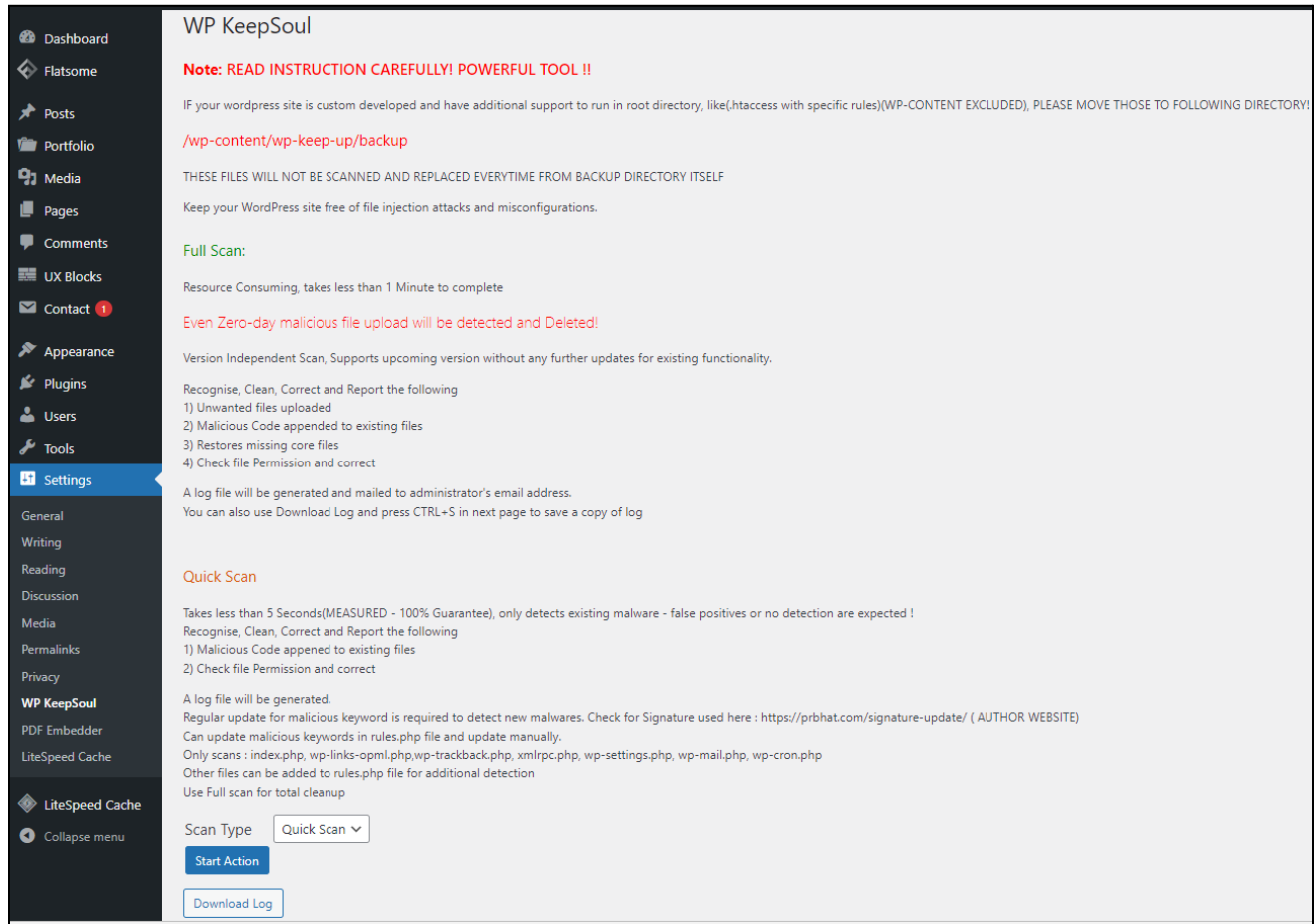


Figure 21. Plugin Page, with instructions and option to perform a scan and download log.

Figure 21 represents the actual page of the plugin. Here, there are four sections:

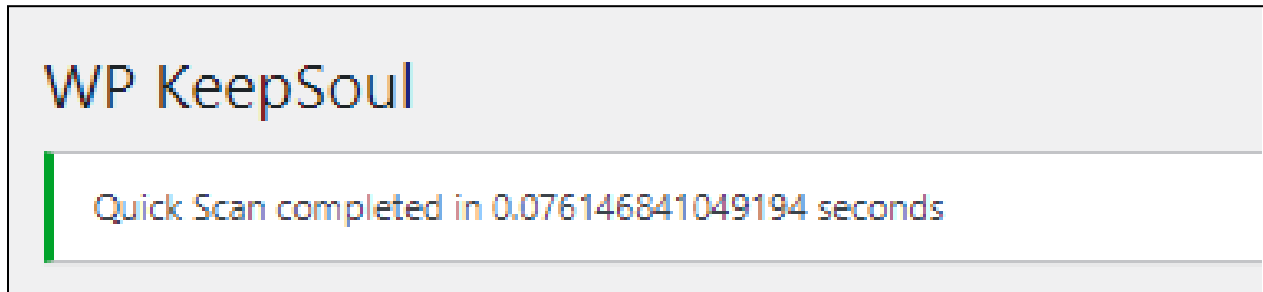
**Section 1:** Heading and mandatory instructions, which has the plugin name and details about the instruction that must be followed before using the plugin. Here, we mention moving the custom-developed files to a backup folder.

**Section 2:** Instructions about the full scan, its working, and output details are mentioned.

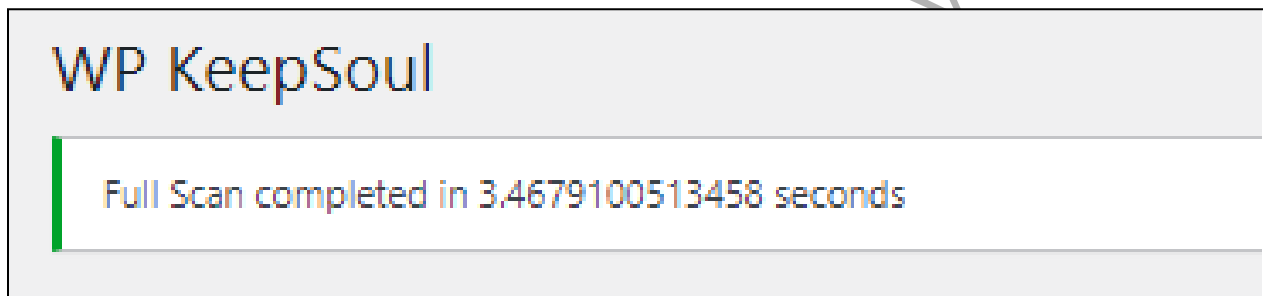
**Section 3:** Details about the quick scan, its limitations to be considered, and working details are mentioned.

**Section 4:** Major part of the plugin that gives access to select scan type and a button to download the log.

All the relevant details mentioned in each section are already described in the methodology modules.



**Figure 22. Notification after completing Quick Scan with the time taken.**



**Figure 23. Notification after completing Full Scan with the time taken.**

Figures 22 and 23 represent simple WordPress notifications with the time taken to complete a particular scan that will appear every time after the scan is complete.

The video demonstrating the complete functioning can be found in Appendix C.

# CHAPTER 6. EVALUATION AND DISCUSSION

## 6.1 Results Evaluation and Limitations Based on Modules

In this section, a thorough and impartial assessment of each module's test results, overall performance, and outcomes are presented. The primary objective of this analysis is to critically evaluate the extent to which all objectives have been met, and the effectiveness of the delivered results of the artefacts. The evaluation will provide valuable insights into the strengths and weaknesses of the modules, which can inform future improvements and developments. Furthermore, it will demonstrate the alignment of the modules with the intended goals and objectives, and the impact of their implementation on the overall project.

### 6.1.1 Module 1: Activation Module

Results from Figures 14 and 15 show the proper control and functioning of the module, which is to download a copy and create necessary directories. Figures 14 and 15 from the results section prove the functioning. However, the plugin requires additional files that add up to 63 megabytes and 3267 inodes which adds to the server resource consumption.

The limitation is that if the hacker takes control of the server and edits the contents in the backup folder, it will be unavoidable to clear malware as the plugin checks in the backup for particular files and restores them without matching them to any other source. One potential limitation of the proposed solution is that if the wp-config.PHP file has been compromised prior to the installation of the plugin, the plugin may not be able to detect the infection. To address this concern, future research could explore the implementation of a quick scan feature as a means of mitigating this issue.

### 6.1.2 Module 2, 3 & 3.1: Full Scan and Quick Scan Module

**Full and Quick scan: Most of the core actions taken and requirements for the proper functioning of both full scan and quick scan are similar and hence, evaluated together.**

The classification of malware used for testing covers almost all types of malware related to WordPress. The full scan, including the mitigation and recovery of the site, takes an average of 9.28222976 seconds, which is among the fastest solutions that include restoration. Even though it detects and restores the compromised files quickly, it will not help in being affected again if a

full-scale attack is performed wipes websites. Table 05 in the methodology section shows the scan results in which three malware were passed through, and the plugin failed to detect them. Another noteworthy point is that the wp-content directory is excluded from scanning. However, this is an area that could be explored further through research and development.

The quick scan module is developed to show another simple and easy way to implement the detection method. Here, the average scan time is 2.073 seconds; however, this method comprises the depth and range of detection and increases dependency as only a set of files with particular keywords are detected using the same functions already present in artefact Module 3.1 (rules.php) further development can be continued with minimal efforts.

Sl. I	Webshell	Type	Full Scan	Time Taken	Comments
83	26.01.2021 - htaccess	Manual	Successful	6.368739128	Full Scan completely cleaned website
84	10.08.2022 - 2index.php	Manual	Successful	6.125956059	Full Scan completely cleaned website
86	15.04.2022 - htaccess	Manual	Successful	6.209491014	Full Scan completely cleaned website
87	09.03.2022 - worker.php	Manual	Successful	6.079495192	Full Scan completely cleaned website
88	28.03.2022 - 1gvyylf.php	Manual	Successful	6.014024973	Full Scan completely cleaned website
90	04.04.2020 - leafmailer.php	Manual	Successful	27.01757097	Full Scan completely cleaned website
<b>Average Time =</b>				<b>9.28222976</b>	

**Figure 24. Snapshot of a full scan, representing the average time of a full scan.** (Full report is in Appendix D)

As critical evaluation suggests, the limitations are only in two modules. They are specified in artefact working condition, and it is a minimum requirement of the plugin to have proper access to the dashboard to perform the scan. Some malware compromises the dashboard, but the functions will still be in working condition in the back end. Hence, the cron job or function URL calling will mitigate the malware in this case. Though, it is impossible to mitigate if the complete dashboard is compromised and scanning cannot be used from Graphical User Interface (GUI) or even a scan function called using URL or cron job; hence, a website cannot be recovered.

Quick scan is introduced for the artefact during the literature review by understanding the JavaScript obfuscation malware and some reported attacks. Limitations are that this method is version dependent and requires frequent updation of keywords in artefact code to perform scan successfully. False detection can also arise if the keywords are not updated. The main reason for implementing the quick scan module is to bring in the scope for future researchers and developers to help upgrade the detection method by using different classified keywords that can help with proper detection. The algorithm for a future redesign of the idea is mentioned in future work.

### 6.1.3 Module 4: Log Module

A logging mechanism records all the tool's incidents. Only a small part that can be considered for evaluation is the creation, deletion and appending to the log.csv file. Initially, the log.csv file will be created while activating the plugin containing entity headers (highlighted in red in Figure 20) that will be used continuously in the logging process. If the administrator deletes the log.csv, the next appending log will not have a header, and only data will be present. Also, the download log button on the admin page needs some modification as it directs to a page with a log that cannot be used directly. Nevertheless, the current logging mechanism gives detailed information about actions taken.

Date	Time	Scan Type	Suspicious Char Count	Original Char Count	File Name	Path	Permission	Action Taken	File Type
12/19/2022	0:03:15	Full Scan	3098657	No backup	.bt	/devmaltest.prbhat.com/public_html/01.12.2020/	.bt	644	Removed
12/19/2022	0:03:15	Full Scan	1742850	No backup	domains.txt	/devmaltest.prbhat.com/public_html/domains.txt		755	Removed
12/19/2022	0:03:15	Full Scan	418	No backup	index.php_	/devmaltest.prbhat.com/public_html/index.php_		755	Removed
12/19/2022	0:03:15	Full Scan	2369	No backup	status.txt	/devmaltest.prbhat.com/public_html/status.txt		755	Removed
12/19/2022	0:03:15	Full Scan	2286	No backup	index.html_	/devmaltest.prbhat.com/public_html/index.html_		755	Removed
12/19/2022	0:03:20	Full Scan	3098657	No backup	.bt	/devmaltest.prbhat.com/public_html/wp-admin/css/	.bt	644	Removed
12/19/2022	0:11:52	Full Scan	1282	2047	admin-post.php	/devmaltest.prbhat.com/public_html/wp-admin/admin-post.php		644	Replaced
12/19/2022	0:15:12	Full Scan	4222	No backup	Coli.Php	/devmaltest.prbhat.com/public_html/Coli.Php		755	Removed
12/19/2022	0:16:49	Full Scan	24593	No backup	sy.php	/devmaltest.prbhat.com/public_html/sy.php		755	Removed
12/19/2022	0:17:34	Full Scan	53480	No backup	bdhrIGW3ARq.php	/devmaltest.prbhat.com/public_html/bdhrIGW3ARq.php		755	Removed
12/19/2022	0:18:43	Full Scan	1793	No backup	cm690ah4.php	/devmaltest.prbhat.com/public_html/cm690ah4.php		755	Removed

Figure 25. Snapshot of scan logs. (Full tabular report is in Appendix G)

It is candidly acknowledged that this issue was identified during the latter stages of the testing phase, and due to time limitations, it was impossible to implement a solution. Nevertheless, the potential resolution for this issue is outlined as an area of focus in future research.

### 6.1.4 Module 5: Email Module

Here, Module 5 is again brought in to give the dimension of future possibilities to developers and researchers while satisfying the essential requirement of email. The results section shows that the mail is received and has all the information about activation and scanning with logs and quarantine files. The email module used is based on the outdated WordPress PHP mailing function and has known security issues; implementing other modules for email would have resulted in major changes to the artefact code and required more time, but the necessary changes are mentioned in future work.

**Overall, all the evaluations and limitations discussed above are unbiased and viewed as a user to understand the critical reflection and to measure future possibilities.**



## CHAPTER 7. CONCLUSION AND FUTURE WORK

Generally, as a whole, this study and artefact help the independent website developers and in-house company web development team, web development agencies, and cyber security investigators to solve the core malware issue within seconds. The usage of resources is also considerably less, and version independency provides better flexibility to operate in multiple sites under the same server without consuming exhausting resources. Using cron jobs will also help mitigate the malware issue by scanning the site as mentioned in the code of the cron job. Considering all the module's functioning, the concept aim, objectives and overall implementation deliver a novel and easy solution to many malware issues in the WordPress platform.

The primary area of future upgradation and scope fall into the quick scan module, email module and inclusion of the wp-content folder and scanning plugins under full scan. The quick scan module uses keywords and characters to detect obfuscated files; it needs to be reassigned for every WordPress release as the code might change during the new version update.

The email module, as discussed, was implemented to bring the possibility inside the artefact, but it needs upgradation from the PHP module to the other latest module; Although this also requires changes in different email variables in all other functions of PHP.

To effectively scan for potential threats, it is recommended to include the wp-content folder in the scanning process. However, this can be challenging as the wp-content directory contains user information, third-party plugins, and externally stored data, making it challenging to map the directory to a separate repository. To address this issue, developers can include the wp-content folder in the scan while testing. A comparison function can be developed in the future with a dataset of all relevant files with character count, which will be utilised to match in a quick scan. If a change is detected, the developer can download the particular file from the backup or relevant sources rather than the entire plugin library. While this may increase scan completion time and use more resources, it provides a dependable solution rather than relying on anti-malware plugins. This artefact helps users, developers, cybersecurity professionals, and companies keep their websites safe and secure.

## CHAPTER 8. APPENDIX LIST

### A. Installation Instruction

1. Download the plugin from the link:  
<https://prbhat.com/wp-content/uploads/2022/12/wp-keepsoul.zip>
2. Upload it to the WordPress website from the dashboard and wait till the upload completes
3. After installation, activate plugin
4. After the activation, navigate to Settings → WP KeepSoul to access plugin
5. Read the instructions and scroll down
6. Choose Quick or Full Scan based on requirement from Dropdown
7. Check the log using the Download Log button.

**B. Source Code Link:** <https://gitlab.uwe.ac.uk/p2-praneethraj/wp-keepsoul>

**C. Walkthrough Video Link:** [https://youtu.be/E\\_WOkIVnIH8](https://youtu.be/E_WOkIVnIH8)

**D. Malware scan report - filtered as full scan with malware classification**

**E. Malware scan report - filtered for quick scan with malware classification.**

**F. Record of supervisor meetings. schedule and notes.**

**G. Scan log from the testing environment - Combined full scan & quick scan.**

## CHAPTER 9. REFERENCES

1. AL-Taharwa, I.A., Lee, H.-M., Jeng, A.B., Wu, K.-P., Mao, C.-H., Wei, T.-E. and Chen, S.-M. (2012) *RedJsod: A Readable JavaScript Obfuscation Detector Using Semantic-based Analysis IEEE Xplore*.1 June 2012 [online]. 1370–1375. Available from: <https://ieeexplore.ieee.org/document/6296140> [Accessed 3 November 2022].
2. Alazab, A., Khraisat, A., Alazab, M. and Singh, S. (2022) Detection of Obfuscated Malicious JavaScript Code. *Future Internet*. [online]. 14 (8), p.217. Available from: <https://www.mdpi.com/1999-5903/14/8/217/pdf> [Accessed 3 September 2022].
3. 'Backdoor.PHP.WEBSHELL.SBJKXRY - Threat Encyclopedia - Trend Micro BE' (2022) *www.trendmicro.com*.19 January 2022 [online]. Available from: <https://www.trendmicro.com/vinfo/be/threat-encyclopedia/malware/backdoor.php.webshell.sbjkxry/> [Accessed 19 December 2022].
4. Blackburn, J. (2022) *WordPress Email Documentation GitHub*.3 November 2022 [online]. Available from: [https://github.com/johnbillion/wp\\_mail](https://github.com/johnbillion/wp_mail) [Accessed 27 December 2022].
5. Cernica, I., Popescu, N. and țigănoaia, B. (2019) *Security Evaluation of Wordpress Backup Plugins IEEE Xplore*.1 May 2019 [online]. 312–316. Available from: <https://ieeexplore.ieee.org/abstract/document/8744951> [Accessed 9 March 2022].
6. 'Clear Command History - Red Team Notes 2.0' (2022) *Gitbook.io*.2022 [online]. Available from: <https://dmcxblue.gitbook.io/red-team-notes-2-0/red-team-techniques/defense-evasion/t1070-indicator-removal-on-host/clear-command-history> [Accessed 6 December 2022].
7. Dalili, S. (2012) *File in the hole secproject.com*.1 November 2012 [online]. HackPra. Available from: <https://soroush.secproject.com/downloadable/File%20in%20the%20hole!.pdf> [Accessed 14 October 2022].
8. Duncan, B. (2015) *InfoSec Handlers Diary Blog SANS Internet Storm Center*.16 September 2015 [online]. Available from: <https://isc.sans.edu/diary/Malicious+spam+with+zip+attachments+containing+.js+files/20153> [Accessed 9 October 2022].
9. Expert, W.S. (2018a) *How to Scan & Detect Malware in WordPress Theme - 2022 WP Hacked Help Blog - WordPress Security Knowledgebase*.17 April 2018 [online].

- Available from:  
<https://secure.wphackedhelp.com/blog/scan-wordpress-theme-for-malware-malicious-code/> [Accessed 1 December 2022].
10. Expert, W.S. (2020) *Spam Links Injection in WordPress Site - Removal Guide [2022] WP Hacked Help Blog - WordPress Security Knowledgebase*. 7 May 2020 [online]. Available from: <https://secure.wphackedhelp.com/blog/spam-link-injection-wordpress/> [Accessed 14 November 2022].
  11. Expert, W.S. (2018b) *WordPress Malware Redirect Hack - How To Fix Guide [2022] WP Hacked Help Blog - WordPress Security Knowledgebase*. 23 May 2018 [online]. Available from: <https://secure.wphackedhelp.com/blog/wordpress-malware-redirect-hack-cleanup/> [Accessed 15 November 2022].
  12. Fang, Y., Huang, C., Su, Y. and Qiu, Y. (2020) Detecting malicious JavaScript code based on semantic analysis. *Computers & Security*. [online]. 93, p.101764.
  13. Fass, A., Backes, M. and Stock, B. (2019) HideNoSeek. *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. [online].
  14. Fass, A., Krawczyk, R.P., Backes, M. and Stock, B. (2018) JaSt: Fully Syntactic Detection of Malicious (Obfuscated) JavaScript. *Detection of Intrusions and Malware, and Vulnerability Assessment*. [online]. pp.303–325.
  15. 'Finding and Removing Spam Links' (2019) *Wordfence*. 16 September 2019 [online]. Available from: <https://www.wordfence.com/learn/removing-spam-links-wordpress/#finding-and-removing-spam-links> [Accessed 26 October 2022].
  16. fm.brianleejackson (2011) *Deep Look Into the WordPress Market Share (2019) Kinsta Managed WordPress Hosting*. 2011 [online]. Available from: <https://kinsta.com/wordpress-market-share/>.
  17. Garand, MacLeod, Martin, Anjos and Sinegubko (2021) *Hacked Websites Trend Report 2021 Sucuri*. 2021 [online]. Available from: <https://sucuri.net/reports/2021-hacked-website-report/> [Accessed 12 November 2022].
  18. Gorji, A. and Abadi, M. (2014) Detecting Obfuscated JavaScript Malware Using Sequences of Internal Function Calls. *Proceedings of the 2014 ACM Southeast Regional Conference*. [online].
  19. Gupta, S. and Gupta, B.B. (2016a) CSSXC: Context-sensitive Sanitization Framework for Web Applications against XSS Vulnerabilities in Cloud Environments. *Procedia Computer Science*. [online]. 85, pp.198–205.

20. Gupta, S. and Gupta, B.B. (2016b) Enhanced XSS Defensive Framework for Web Applications Deployed in the Virtual Machines of Cloud Computing Environment. *Procedia Technology*. [online]. 24, pp.1595–1602.
21. He, X., Xu, L. and Cha, C. (2018) *Malicious JavaScript Code Detection Based on Hybrid Analysis IEEE Xplore*.1 December 2018 [online]. 365–374. Available from: <https://ieeexplore.ieee.org/document/8719574> [Accessed 3 November 2022].
22. hungered (2009) *Secure File Upload Check List With PHP Hungred Dot Com*.17 August 2009 [online]. Available from: <https://www.hungred.com/useful-information/secure-file-upload-check-list-php/> [Accessed 14 October 2022].
23. Investigation, F.B. (2016) *GRIZZLY STEPPE -Russian Malicious Cyber Activity* [online]. Available from: [https://www.cisa.gov/uscert/sites/default/files/publications/JAR\\_16-20296A\\_GRIZZLY%20STEPPE-2016-1229.pdf](https://www.cisa.gov/uscert/sites/default/files/publications/JAR_16-20296A_GRIZZLY%20STEPPE-2016-1229.pdf).
24. junaidtk (2020) *How to create a basic Cron job scheduling in WP Gist*.2020 [online]. Available from: <https://gist.github.com/junaidtk/941c9cb8b1b12eab2d90014a85d50a45> [Accessed 27 December 2022].
25. kamermans (2015) *PHP time() vs microtime() vs microtime(true) benchmark Gist*.2015 [online]. Available from: <https://gist.github.com/kamermans/04d3e557efca29bf2123> [Accessed 27 December 2022].
26. Karl (2017) *A Thousand Monkeys Writing a JavaScript Malware Downloader: De-obfuscating the JavaScript Malware Musings*.18 January 2017 [online]. Available from: <https://malwaremusings.com/2017/01/19/a-thousand-monkeys-writing-a-javascript-malware-downloader-de-obfuscating-the-javascript/> [Accessed 21 October 2022].
27. Kasturi, R.P., Fuller, J., Sun, Y., Chabklo, O., Rodriguez, A., Park, J. and Saltaformaggio, B. (2022) *Mistrust Plugins You Must: A {Large-Scale} Study Of Malicious Plugins In {WordPress} Marketplaces www.usenix.org.2022* [online]. 161–178. Available from: <https://www.usenix.org/conference/usenixsecurity22/presentation/kasturi> [Accessed 10 November 2022].
28. Kino, K. (2021) *PHP Malware Used in Lucky Visitor Scam JPCERT/CC Eyes*.4 June 2021 [online]. Available from: [https://blogs.jpccert.or.jp/en/2021/06/php\\_malware.html](https://blogs.jpccert.or.jp/en/2021/06/php_malware.html) [Accessed 20 December 2022].

29. KL, A. (2022) *What Is Arbitrary File Upload Vulnerability? How To Protect Form It? The Sec Master*.12 April 2022 [online]. Available from: <https://theseckmaster.com/what-is-arbitrary-file-upload-vulnerability-how-to-protect-form-it/> [Accessed 20 October 2022].
30. 'Kovter & Miuref/Boaxxe Infections | Inside a Malspam Operation' (2015) <https://cofense.com/>.11 September 2015 [online]. Available from: <https://cofense.com/a-peek-inside-an-affiliates-malspam-operation-kovter-and-miurefboaxxe-infections/> [Accessed 9 October 2022].
31. Livshits, C.C.B., Zorn, B. and Seifert, C. (2010) *Zozzle: Low-overhead Mostly Static JavaScript Malware Detection* [www.microsoft.com](http://www.microsoft.com).26 November 2010 [online]. Available from: <https://www.microsoft.com/en-us/research/publication/zozzle-low-overhead-mostly-static-javascript-malware-detection/>.
32. luke (2020) *SCP-173 PHP Malware + WordPress | Cybersecurity Research SCP-173 PHP Malware + WordPress*.12 October 2020 [online]. Available from: <https://lukeleal.com/research/posts/scp-173-malware/> [Accessed 19 December 2022].
33. marcosnakamine (no date) *WordPress - Send mail wp\_mail with attachment* *Gist*. [online]. Available from: <https://gist.github.com/marcosnakamine/9443303c340ef30ee3bac0f6d8eac95b> [Accessed 27 December 2022].
34. Md, M.H., Sarker, K., Biswas, S. and Md, H.S. (2017) Detection of Wordpress Content Injection Vulnerability. *International Journal on Cybernetics & Informatics*. [online]. 6 (5), pp.1–15.
35. Mesa, O., Vieira, R., Viana, M., Durelli, V.H.S., Cirilo, E., Kalinowski, M. and Lucena, C. (2018) Understanding vulnerabilities in plugin-based web systems. *Proceedings of the 22nd International Conference on Systems and Software Product Line - SPLC '18*. [online].
36. Milan (2021) *Minimum WordPress Server Requirements WP Thinker*.20 September 2021 [online]. Available from: <https://wpthinker.com/minimum-wordpress-server-requirements/> [Accessed 24 December 2022].
37. Moog, M., Demmel, M., Backes, M. and Fass, A. (2021) *Statically Detecting JavaScript Obfuscation and Minification Techniques in the Wild IEEE Xplore*.1 June 2021 [online].

- 569–580. Available from: <https://ieeexplore.ieee.org/document/9505063> [Accessed 27 September 2022].
38. Moran, M. (2022) *WordPress Hacking Statistics (How Many Websites Get Hacked?) Colorlib*.23 October 2022 [online]. Available from: <https://colorlib.com/wp/wordpress-hacking-statistics/> [Accessed 25 December 2022].
  39. OWASP, Dalili, S., Wetter, D. and Mayo, L. (2020) *Unrestricted File Upload | OWASP owasp.org*.5 September 2020 [online]. Available from: [https://owasp.org/www-community/vulnerabilities/Unrestricted\\_File\\_Upload](https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload) [Accessed 10 August 2022].
  40. Patil, D.R. and Patil, J.B. (2017) Detection of Malicious JavaScript Code in Web Pages. *Indian Journal of Science and Technology*. [online]. 10 (19), pp.1–12.
  41. Pejic, S. (2022a) *Malware Scanner cPanel plugin GitHub*.12 October 2022 [online]. Available from: <https://github.com/stefanpejic/malware-scanner-cpanel-plugin> [Accessed 10 December 2022].
  42. Pejic, S. (2022b) *Simple Virus Scanner cPanel plugin GitHub*.12 October 2022 [online]. Available from: <https://github.com/stefanpejic/simple-virus-scanner-cpanel-plugin> [Accessed 10 December 2022].
  43. Pejic, S. (2022c) *WordPress Malware GitHub*.6 December 2022 [online]. Available from: <https://github.com/stefanpejic/wordpress-malware> [Accessed 10 December 2022].
  44. Petrak, H. (2022) *Javascript Malware Collection GitHub*.26 November 2022 [online]. Available from: <https://github.com/HynekPetrak/javascript-malware-collection> [Accessed 27 November 2022].
  45. ‘php.hacktool.mailer.010’ (2019) *Sucuri Labs*.1 April 2019 [online]. Available from: <https://labs.sucuri.net/signatures/malwares/php.hacktool.mailer.010/> [Accessed 19 December 2022].
  46. portswigger (no date) *File uploads | Web Security Academy portswigger.net*. [online]. Available from: <https://portswigger.net/web-security/file-upload>.
  47. Rees, K. (2022) *Over 15,000 WordPress Sites Affected in Malicious SEO Campaign MUO*.15 November 2022 [online]. Available from: <https://www.makeuseof.com/15000-wordpress-sites-affected-in-malicious-seo-campaign/> [Accessed 9 December 2022].
  48. Ročevs, Š. (2021) *7 SDKs to Send Emails in Your Favorite Programming Language MailerSend*.23 September 2021 [online]. Available from:

<https://www.mailersend.com/blog/send-emails-in-your-favorite-programming-language>  
[Accessed 23 December 2022].

49. 'Scan' (no date) *Wordfence*. [online]. Available from: <https://www.wordfence.com/help/scan/> [Accessed 23 December 2022].
50. Shah, A. (2016) Malicious JavaScript Detection using Statistical Language Model. *Master's Projects*. [online]. Available from: [https://scholarworks.sjsu.edu/etd\\_projects/476/](https://scholarworks.sjsu.edu/etd_projects/476/) [Accessed 10 November 2022].
51. Sohan, Md.F. and Basalamah, A. (2020) A Systematic Literature Review and Quality Analysis of Javascript Malware Detection. *IEEE Access*. [online]. 8 (2169–3536), pp.190539–190552.
52. Su, J., Yoshioka, K., Shikata, J. and Matsumoto, T. (2016) An Efficient Method for Detecting Obfuscated Suspicious JavaScript Based on Text Pattern Analysis. *Proceedings of the 2016 ACM International on Workshop on Traffic Measurements for Cybersecurity*. [online].
53. Tuca, A. (2022) *Which Type of Website Can Be Built Using WordPress? 10 Ideas for 2022 Themeisle Blog.2* May 2022 [online]. Available from: <https://themeisle.com/blog/which-type-of-website-can-be-built-using-wordpress/> [Accessed 25 December 2022].
54. Ullrich, J. (2009) *SANS Institute | 8 Basic Rules to Implement Secure File Uploads | SANS Institute www.sans.org*.28 December 2009 [online]. Available from: <https://www.sans.org/blog/8-basic-rules-to-implement-secure-file-uploads/> [Accessed 14 October 2022].
55. 'VirusTotal' (no date) *www.virustotal.com*. [online]. Available from: <https://www.virustotal.com/gui/file/61d702610d6ae9a0a494695186843ac2d7e2aea19b8417d6e7961e1a9b648e3b> [Accessed 24 December 2022].
56. W3Techs (no date) *Usage Statistics and Market Share of Content Management Systems w3techs.com*. [online]. Available from: [https://w3techs.com/technologies/overview/content\\_management](https://w3techs.com/technologies/overview/content_management).
57. 'What is WordPress? A Beginner's Guide (FAQs + Pros and Cons)' (2020) *www.wpbeginner.com*.19 August 2020 [online]. Available from: <https://www.wpbeginner.com/beginners-guide/what-is-wordpress/>.
58. Williams, L., Massacci, F. and Gary (2021) *Secure Software Lifecycle Knowledge Area Version* [online]. Available from:



[https://www.cybok.org/media/downloads/Secure\\_Software\\_Lifecycle\\_v1.0.2.pdf](https://www.cybok.org/media/downloads/Secure_Software_Lifecycle_v1.0.2.pdf)  
[Accessed 25 December 2022].

59. Wilson, U. (2022) *devuri/wp-admin-page* *GitHub*. 10 February 2022 [online]. Available from: <https://github.com/devuri/wp-admin-page> [Accessed 27 December 2022].
60. 'WordPress Malware Removal Service - No Wait Time, 1 Click Malware Cleaner' (no date) *Malcare*. [online]. Available from: <https://www.malcare.com/wordpress-malware-removal/>.
61. 'wp\_mail() | Function' (no date) *WordPress Developer Resources*. [online]. Available from: [https://developer.wordpress.org/reference/functions/wp\\_mail/](https://developer.wordpress.org/reference/functions/wp_mail/) [Accessed 9 December 2022].
62. 'wpmudev/p3-profiler' (2022) *GitHub*. 18 October 2022 [online]. Available from: <https://github.com/wpmudev/p3-profiler> [Accessed 10 December 2022].
63. Wu, H. and Qin, S. (2017) *Detecting obfuscated suspicious JavaScript based on collaborative training IEEE Xplore*. 1 October 2017 [online]. 1962–1966. Available from: <https://ieeexplore.ieee.org/document/8359972> [Accessed 3 November 2022].



PRANEETHRAJ WP K