

Plugging the Gaps: Investigating the Causes of Patch Delays and Suspensions in WordPress Plugin Development

Oliver Toparti

10135169

Supervised by

Dr Ian Cornelius

School of Computing, Mathematics and Data Science

21st April 2023



6047CEM Declaration of Originality

I declare that this project is all my own work and has not been copied in part or in whole from any other source except where duly acknowledged. As such, all use of previously published work (from books, journals, magazines, internet etc) has been acknowledged by citation within the main report to an item in the References or Bibliography lists. I also agree that an electronic copy of this project may be stored and used for purposes of plagiarism prevention and detection.

Statement of Copyright

I acknowledge that the copyright of this project report, and any product developed as part of the project, belong to Coventry University. Support, including funding, is available to commercialise products and services developed by staff and students. Any revenue that is generated is split with the inventor(s) of the product or service. For further information, please see www.coventry.ac.uk/ipr or contact ipr@coventry.ac.uk.

Statement of Ethical Engagement

I declare that a proposal for this project has been submitted to the Coventry University ethics monitoring website (<https://ethics.coventry.ac.uk>) and that the application number is listed below.

Note: Projects without an ethics application number will be rejected for marking.

Signed: Oliver Toparti

Date: 21st April 2023

First Name	Oliver
Last Name	Toparti
Student ID	10135169
Ethics Application Number	P-147828
First Supervisors Name	Dr Ian Cornelius

Abstract

The subsequent dissertation investigates the causes and impacts of patch delays and suspensions in WordPress plugin development, as well as their effects on users'. With over 93% of WordPress vulnerabilities originating from plugins, this dissertation aims to provide a comprehensive understanding of the issues from both the developer and user perspectives. To this end, the paper analyses contemporary literature, carries out qualitative research, and designs and implements a proof-of-concept software solution called the WordPress Plugin Patch (WPP) Monitor plugin. The paper's findings highlight the importance of monitoring WordPress plugin patches, with the WPP Monitor plugin incorporating an automatic notification system to ensure timely patching. Overall, the study's findings highlight the importance of monitoring WordPress plugin patches, with the WPP Monitor plugin incorporating an automatic notification system to ensure timely patching. The dissertation's findings reveal that only 21.7% of small-to-medium sized enterprises (SMEs) have an advanced understanding of web development, leading over 52% of them to outsource web development to third-party developers. Furthermore, concludes that many web developers make mistakes that leave SMEs vulnerable, such as selecting plugins based solely on ratings and reviews and installing ineffective security plugins. Despite limited research in this area, the dissertation aims to fill this gap.

Contents

1.	Introduction.....	10
1.1	Background.....	10
1.2	Research Question	11
1.3	Aims and Objectives.....	11
1.4	Research Structure.....	11
2.	Literature Review	12
2.1	Plugin Vulnerabilities	13
2.1.1	Frequency.....	13
2.1.2	Complexity.....	14
2.1.3	Relevance.....	15
2.2	Current Solutions for Reducing Risks	15
2.2.1	Efficacy.....	16
2.2.2	Accessibility.....	16
2.2.3	Compatibility	17
2.3	Impact of Security Patch Delays.....	17
2.3.1	Case Study 1: Equifax Data Breach.....	17
2.3.2	Case Study 2: WPGateway Zero-day Exploit.....	18
2.4	Summary.....	18
3.	Methodology.....	20
3.1	Web Developers.....	20
3.2	Plugin Developers.....	22
3.3	WordPress Users.....	23
3.4	Tools and Requirements	24
3.4.1	Functional Requirements	24
3.4.2	Non-Functional Requirements	25
3.5	Concept Software.....	26
3.5.1	Design	26
3.5.2	Testbed.....	26
3.6	Data Collection and Management	27
4.	Implementation.....	28
4.1	Manager	28
4.2	Engine.....	31
5.	Results and Analysis.....	33
5.1	Web Developers Survey	33
5.2	Plugin Developers Survey	34
5.3	WordPress Users Survey	35
5.4	Concept Software.....	38
6.	Discussion.....	39

7.	Project Management	40
7.1	Agile Software Methodology	40
7.2	Legal, Ethical and Social Constraints	40
7.3	Student Reflection	41
7.4	Initial Gant Chart	42
7.5	Post Gant Chart.....	43
8.	Conclusion	44
8.1	Future Work.....	45
	Appendices	48
Appendix A	Data Extractor.....	48
Appendix B	Concept Software	49

List of Tables

Table 2.1 Most relevant developer discussions (Murphy et al., 2021).....	15
Table 3.1 Web developers survey questions.....	20
Table 3.2 Plugin developers survey questions.....	22
Table 3.3 WordPress users survey questions.....	23
Table 5.1 The concept software's efficacy.	38
Table 8.1 Measuring the aims and objectives.....	44

List of Listings

Listing 4.1 Dashboard code.	28
Listing 4.2 Active scan tab code.....	29
Listing 4.3 Settings tab code one.....	30
Listing 4.4 Settings tab save or scan code.	30
Listing 4.5 Integrations tab code.	31
Listing 4.6 Scan function code.	31
Listing 4.7 Email function code.....	32

List of Figures

Figure 2.1 Most frequently reported plugin vulnerabilities (Mesa et al., 2018).....	13
Figure 2.2 Most frequently reported plugin vulnerabilities repeated.	14
Figure 2.3 Efficacy comparison of popular WordPress security plugins (Murphy et. al., 2021).....	16
Figure 2.4 Visual representation of the research problem.	17
Figure 3.1 UML interaction diagram.....	25
Figure 3.2 WPP Monitor Plugin functionality overview.....	26
Figure 4.1 Example of the agile development testing phase	32
Figure 5.1 Pie chart: How many employees does your business have?	36
Figure 5.2 Pie chart: Do you outsource your WordPress web development?	36
Figure 5.3 Pie chart: How would you describe your WordPress web development skills?	36
Figure 5.4 Pie chart: Have you ever experienced a security incident that has affected your website?	37
Figure 5.5 Pie chart: How do you communicate with your developer regarding website security?	37
Figure 7.1 Overview of the agile software development cycle stages	40
Figure 7.2 Initial Gantt chart of project tasks and milestones.	42
Figure 7.3 Post Gantt chart of project tasks and milestones.....	43

List of Abbreviations

Apache, Linux, MySQL, and PHP (LAMP)	17
Common Vulnerability and Exposure (CVE)	11
Content Management System (CMS)	11
Cross-Site Request Forgery (CSRF)	15
Cross-Site Scripting (XSS)	11, 14, 15
Directory Traversal Attacks (DTA)	11
Dormant Pattern (P3)	14
Institute of Electrical and Electronics Engineers (IEEE)	13
JavaScript Object Notation (JSON)	14, 15
National Vulnerability Database (NVD)	14, 15
Occasional Pattern (P2)	14
Persistent Pattern (P1)	14
Prevalent Pattern (P4)	14
Small and medium-sized enterprise (SME)	11
Small and medium-sized enterprises (SMEs)	24
Structured Query Language Injection (SQLi)	11
Student Reflection (SR)	41
The Association for Computing Machinery (ACM)	13
WordPress Plugin Patch (WPP)	12

1. Introduction

1.1 Background

In 2013, WordPress became the leading Content Management System (CMS) of the modern web, now powering more than 43.2% of all websites on the internet (W3Techs, 2023). The framework was created as an open-source project in 2003 by Matt Mullenweg and Mike Little with the intention to make blogging more user-friendly for non-technical users (Cabot, 2018). It has gained widespread attention in 2004 with the introduction of its plugin architecture, which allows users to extend the framework's functionality by writing their own plugins and sharing them with the community (Mullenweg, 2004). Today, the WordPress plugin directory stores over 60,000 plugins, while marketplaces like Themeforest and CodeCanyon offer thousands for a one-off or monthly cost (Kasturi et al., 2022).

Despite its modern-day success, WordPress was found to be the most commonly hacked CMS in 2021 with thousands of attacks occurring every minute (Gall, 2023; Sucuri, 2021). Although the specific categories of web exploitation may change annually, they usually encompass Cross-Site Scripting (XSS), Structured Query Language Injection (SQLi), and Directory Traversal Attacks (DTA) (Gall, 2023). As a result of this, website owners encounter various security incidents, from data breaches to website defacement, downtime, and reputation damage. In a recent WordPress security report, researchers found that 93.2% of the vulnerabilities disclosed in 2022 originated from plugins (Knauss, 2022). Furthermore, 26% of plugins were left unpatched at the time of vulnerability disclosure, and 5% were suspended from the WordPress plugin directory (Knauss, 2022). These figures highlight two critical security issues:

1. Plugin vulnerabilities are often disclosed in Common Vulnerability and Exposure (CVE) databases. If plugin vulnerabilities are publicised faster than patches are released, it leaves a hiatus for cybercriminals to exploit websites.
2. Users do not get notified of plugins which have been suspended from the WordPress directory. In fact, *“users need to manually check if they have suspended plugins installed and remove them”* (Patchstack, 2022).

These two security issues are closely interconnected. WordPress informs developers if a vulnerability is reported by a researcher (Patchstack, 2022) and if developers fail to address vulnerabilities in a timely manner, their plugins may be suspended (Williams et al., 2020).

1.2 Research Question

The aim of this research is to answer, “*What are the causes and how can users reduce the risks of becoming vulnerable from WordPress plugin patch delays and suspensions?*”. The complexity of the question makes it difficult to provide a straightforward answer. As a result, it may be helpful to divide it into smaller, more manageable subsets of questions, such as the following:

- RQ1. What are the primary causes of plugin patch delays and suspensions?
- RQ2. Who is the responsible party for plugin patch delays and suspensions?
- RQ3. What steps can be taken, from the viewpoints of both users and developers, to mitigate risks?

1.3 Aims and Objectives

To answer these multifaceted questions, it is essential to understand the problem from the two different perspectives: developers and users. A user is defined as any entity or organisation that identifies itself as Small and medium-sized enterprise (SME) and makes use of any type of WordPress plugins. Contrarily, a developer is defined as any individual or organisation that creates and distributes WordPress Plugins. The following objectives are outlined to stay within the scope of the project.

1. Identify the causes of patch delays and plugin suspensions in WordPress Plugin Development and critically evaluate their impact in previous cases.
2. Carry out qualitative research on experienced individuals or business representatives to compose an analytical report.
3. Design and implement a proof-of-concept software that solves parts or the entire problem, meeting the demographic group criteria.

There are two types of WordPress version including the open-source version and the service version. It is important to note that this research will only focus on the open-source version and the anything else will be out of the scope and therefore won't be included in the research.

1.4 Research Structure

The following report aims to provide an in-depth analysis of the causes and impacts of patch delays and suspensions in WordPress plugin development. It proposes a software solution called the WordPress Plugin Patch (WPP) Monitor plugin, which incorporates an automatic notification system for monitoring WordPress plugin patches. The report is structured into several sections, starting with a literature review that analyses and critiques contemporary literature to establish a theoretical and technical foundation for the research question. The methodology section provides an in-depth overview of the research methodology to ensure consistency, validity, and transparency, focusing on three distinct demographic groups: web developers, plugin developers, and WordPress users. The implementation chapter offers a comprehensive overview of the development process for the WPP Monitor plugin, following the agile methodology to ensure efficient and timely completion of the project. The subsequent results and analysis section provides insights into the findings of the data collection outlined in the methodology section. The discussion section critically evaluates the study's findings and how they fit into current studies, while the project management section outlines the methodologies employed in the project to establish its validity. Lastly, the conclusion section summarises the initial objectives, the findings of the project, and outlines potential future work.

2. Literature Review

The following section analyses and critiques contemporary literature, to establish a theoretical, and technical foundation for the research question (refer to Section 1.2). Finding appropriate literature for this multifaceted question requires to use search terms and sources which are not directly correlated to WordPress; however, they are still considered to be in the scope of the review. Sources in the literature review are extracted from Google Scholar, Institute of Electrical and Electronics Engineers (IEEE) Xplore, Science Direct, and The Association for Computing Machinery (ACM) Digital Library. It is analysed using thematic analysis to identify common themes and patterns related to the causes and impacts of patch delays and suspensions in WordPress plugin development.

Section 2.1 explores the fundamentals of plugin vulnerabilities to gain insights into the frequency, complexity, and relevance of this issue. One common misconception is that complex vulnerabilities require more time and effort to address (Mesa et al., 2018). However, without context or statistical evidence, it is difficult to prove this correlation. To debunk this myth, this section critiques a 2018 study which aimed to identify the most frequent plugin vulnerabilities, the complexity of patching plugin vulnerabilities, and the relevance of plugin vulnerability-related discussions among developers (Mesa et al., 2018). Given that this study is now half a decade old, several parts of this study is repeated and compared with the initial researchers' findings, to assess how vulnerabilities have evolved over the years. This review is significant to the research as studying the vulnerabilities will help establish a strong technical foundation of the intricate tasks that WordPress plugin developers perform. As a result, this can provide insights into the underlying causes of patch delays and suspensions from a developer's perspective.

Section 2.2 provides an in-depth analysis of the current-day WordPress security plugins to highlight the gap of vulnerability detection solutions. This section incorporates a 2021 study in which current-day solutions that claimed to *"reduce the risk of becoming vulnerable from plugin vulnerabilities"* were tested in three distinct areas: efficacy, accessibility, and compatibility (Murphy et al., 2021). Understanding the strengths and weaknesses of these security plugins helps to identify the gaps and limitations of current mitigation tools. As a result, this can provide insights into the underlying causes of patch delays and suspensions from a user's perspective.

Section 2.3 explores the effects of patch delays on end-users and discusses the roles of the three key players involved. The section also provides justification for the involvement of a third-party in addressing this issue. There are two case studies that are covered in this section, one of which is the 2017 Equifax breach. This breach was caused by a delay in patching a critical vulnerability in a web application, which was addressed several months after the vulnerability was disclosed (Berghel, 2017). While this case is not in direct correlation with WordPress, the consequences of patch delay can be similarly severe, therefore this study is considered to be within scope. The second case study, WPGateway zero-day vulnerability, occurred in 2022. This vulnerability scored 9.8 and impacted 280,000 websites, with the disclosure happening faster than the flaw was patched (Gall, 2022). Exploring this case study will better understand how the vulnerability has impacted end-users. As a result, this can provide insights into the underlying impacts of patch delays and suspensions from user's perspective.

2.1 Plugin Vulnerabilities

The fundamentals of plugin vulnerabilities can be broken down into three distinct areas: frequency, complexity, and relevance. In 2018, Mesa et al. (2018) has conducted an exploratory study on WordPress plugins with the objective to identify the most frequent plugin vulnerabilities, the complexity of patching plugin vulnerabilities, and the relevance of plugin vulnerability-related discussions among developers. The following section critiques this study and compares it with similar research of this area, highlighting the gaps.

2.1.1 Frequency

To determine the most frequently encountered plugin vulnerability types, Mesa et al. (2018) has extracted disclosed vulnerability records from the National Vulnerability Database (NVD). The extraction process involved downloading a complete set of vulnerability records in JavaScript Object Notation (JSON) format and loading it into a parsing software to filter out the descriptions that included the terms “WordPress” and “Plugin.” After parsing the JSON files, 895 matching records were found. Based on the matching records, the Common Weakness Enumeration Specification (CWE) were also extracted and were categorised into their corresponding vulnerability types. The graph shown in Figure 2.1, highlights the results of this study.

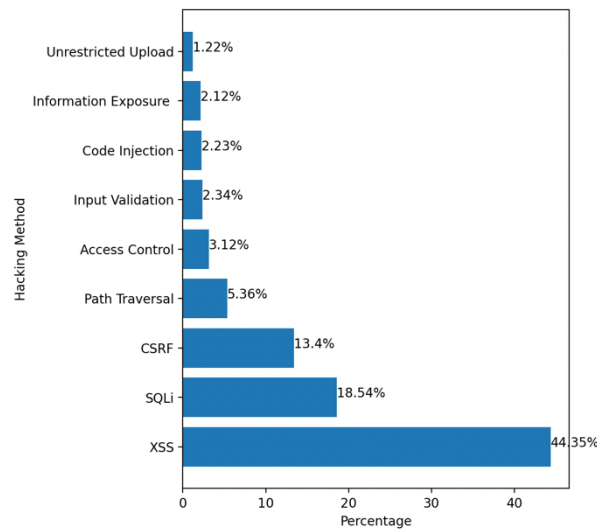


Figure 2.1 Most frequently reported plugin vulnerabilities (Mesa et al., 2018).

The results of this study indicate that the most frequently encountered vulnerabilities between 2006 and 2016 were XSS, SQLi and CSRF. In addition to this study, Mesa et al. (2018) has also analysed the annual frequency of vulnerabilities and identified four evolution patterns, namely persistent (P1), occasional (P2), dormant (P3), and prevalent (P4). According to Mesa et al. (2018), vulnerabilities caused by plugins tend to follow a persistent evolution pattern, with 76.31% of the vulnerabilities repeatedly reoccurring over the years (Mesa et al., 2018). XSS and SQLi are the common types of vulnerabilities that fall into this pattern. The only vulnerability type identified for P2 was the Management of Permissions. While vulnerabilities related to permission management were observed sporadically in the past, the study suggests, that there was a surge in the frequency of this type of vulnerability between 2006 and 2016. Code injection was also the only vulnerability to fall under P3.

Despite that this study is half a decade old, the results still indicate the kind of vulnerabilities that can “make users vulnerable if plugin developers do not acknowledge the importance of secure programming such as eval().” (Mesa et al., 2018). To support this statement even further, it is essential to replicate this research to assess how vulnerabilities have evolved. Unfortunately, the research methodology of this study, lacks clear detail which makes it necessary to make several assumptions. Therefore, it is assumed that the parser software utilised in the study was written in Python, based on a simple script (refer to Appendix A). This analysis utilises JavaScript Object Notation (JSON) vulnerability records sourced from the National Vulnerability Database (NVD), with a focus on the

records from 2006 to January 2023. Figure 2.2 depicts the most commonly reported vulnerabilities between 2002 and 2023, according to the NVD.

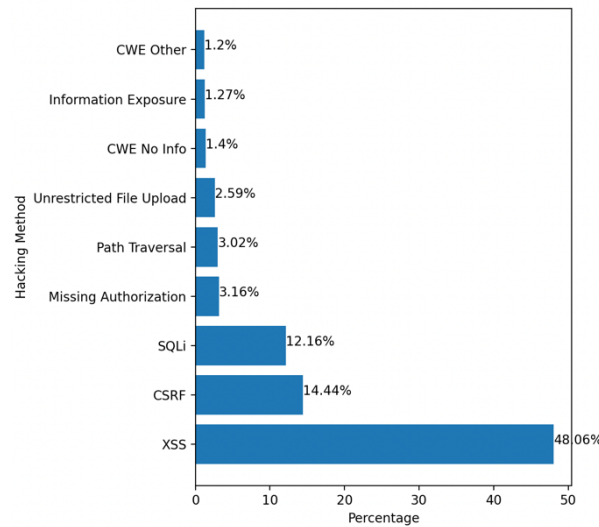


Figure 2.2 Most frequently reported plugin vulnerabilities repeated.

Based on a replicated study, the most common WordPress plugin vulnerabilities encountered between 2006 and 2023 were XSS, SQLi and Cross-Site Request Forgery (CSRF). These findings are consistent with the results reported by Aborujilah et al., 2022, who found that the most frequent vulnerabilities in web applications are XSS, SQLi, and CSRF. Thus, it appears that WordPress plugin developers continue to face the same challenges as identified in the initial study, indicating that despite technological advancements, end-users are still vulnerable to these exploitations. Furthermore, these findings support the conclusions drawn in Mesa et al. (2018) repeated research that vulnerabilities falling under P1 persistently occurs over time and are likely to consistently appear in current vulnerability rankings. Therefore, plugin developers should be particularly cautious of vulnerabilities falling under P1, such as XSS, SQLi, and CSRF, as they have a high likelihood of remaining prevalent and recurring in the future.

2.1.2 Complexity

To ascertain the complexity involved in patching vulnerabilities, Mesa et al. (2018) examined 109 patch reports from the 895 previously analysed bulletins. These reports contain source code modifications aimed at fixing or improving WordPress plugins. The authors found that 42% of the patches required adding new lines of code, 67% required changing existing lines of code, and 18% required removing code altogether. Although some patches did involve removing code, most of the changes were straightforward, such as calling a sanitisation method. Analysis of the 119 patches studied revealed that fixing a vulnerability typically only required changes to a single file, with most changes focused on only a few files. In the worst-case scenario, up to 18 files required modification.

The three largest patches examined in the study involved sanitising a large volume of parameters and creating supporting procedures to prevent SQLi. For instance, the patch for fixing multiple SQLi vulnerabilities in the “Simple Ads Manager” plugin required modifying 35 lines of code across two files. Similarly, fixing two vulnerabilities in the “Video Embed & Thumbnail Generator” plugin involved changing 72 lines of code through the implementation of a media upload form and Generator functionalities, which were to aid in the processing of input data.

In conclusion, the study suggests that while patch sizes for post-release security vulnerabilities may be significant in certain cases, most changes tend to be relatively simple and focused on only a few files. Mesa et al. (2018) believes given the nature of most common vulnerabilities, it is clear that developers do not understand common attacks and how easy mitigation methods can avoid complex security problems. It is advised that inexperienced developers would need tools that would enable them to build reliable and secure plugins. From this research it can be concluded that developers prioritise

functionality over the security of plugin users. It is therefore important to consider that if plugin developers are not able to develop secure plugins, then how can users avoid becoming vulnerable.

2.1.3 Relevance

To identify the relevance of vulnerability-related discussions among developers, Mesa et al. (2018) analysed the WordPress Stack Exchange forum to determine the most common security-related topics among plugin developers. The authors extracted questions and answers from the forum using the Stack Exchange application programming interface (API), covering the period between August 2010 and October 2017. The researchers identified security-related questions by searching for keywords such as "vulnerability," "hacked," and "security" in the titles and bodies of the questions. To ensure the relevance of their analysis, Mesa et al. (2018) removed code from the answers and questions. The researchers used the Latent Dirichlet Allocation (LDA) technique to extract, calculate and assign a score to the most common security topics discussed amongst developers. To ensure the topics properly captured the general, perceived, and potential security concerns discussed by WordPress developers, the researchers chose more coarse-grained topics and assigned a severity value for each topic. Table 2.1 shows the topics identified.

Table 2.1 Most relevant developer discussions (Murphy et al., 2021).

Topics Name	Q	A	Q/A	Views	Score
Server Security	11	19	1.72	440.09	5.09
Cryptography	38	54	1.42	991.02	1.84
Authentication	43	59	1.37	1947.55	1.37
Plugin Update	27	33	1.22	386.77	1.37
Permissions	34	51	1.5	1801.32	1.32
File Upload	29	36	1.24	1835.62	1.93
Content Security	43	60	1.39	855.04	1.16
Programming	50	63	1.26	517.04	2.4
SSL Certificate	42	45	1.07	1209.28	0.4
Database Security	291	410	1.63	794.16	1.63

According to the findings of Mesa et al. (2018), database security was the most frequently queried vulnerability topic among developers. Despite having a relatively low view count, the number of questions and answers related to this topic was almost six times higher than those related to content security or programming. It can be inferred that the reason behind the significant number of queries related to database security could be linked to previous findings on SQLi, which is a prevalent vulnerability in current times falling under P1. Conversely, it can be speculated that server security has the lowest number of questions and answers because WordPress plugin developers typically do not handle server security, as it falls under the responsibility of the hosting company. Despite having a lower number of questions and answers, server security has a higher score than most topics due to its significance and the potentially high losses resulting from successful attacks.

According to Mesa et al. (2018), “*despite the low number of questions, we believe that Programming is the most important topic to guide the overall progress towards proactive security*” (Mesa et al. 2018). This statement is quite general but valid as without proper programming techniques and practices, applications can be vulnerable to a variety of security threats, including SQLi, XSS, and other types of attacks.

2.2 Current Solutions for Reducing Risks

Web-based security plugins are one of the most commonly utilised solutions among WordPress users, aiming to reduce potential vulnerabilities discussed (refer to Section 2.1). Their capability can be

evaluated in efficacy, accessibility, and compatibility. This section provides an in-depth analysis of current-day WordPress security plugins to highlight the gap of vulnerability detection solutions.

2.2.1 Efficacy

Efficacy refers to the ability to accurately identify and mitigate potential security vulnerabilities (refer to Section 2.1). In 2021, Murphy et al. (2021) has conducted an empirical assessment on WordPress security plugins with the objective to assess efficacy. To conduct the research, two sample selection processes were adopted. Firstly, the security plugin selection, involved the criteria of selecting the eleven most popular and freely available plugins, that “*claimed to address security issues.*”. Secondly, the vulnerable plugins selection, involved the selection of 51 feature-rich vulnerable plugins from the Exploit Database. To ensure minimal bias, the researchers have set up an Apache, Linux, MySQL, and PHP (LAMP) environment and independently tested the efficacy of the eleven security plugins against the fifty-one feature-rich vulnerabilities.

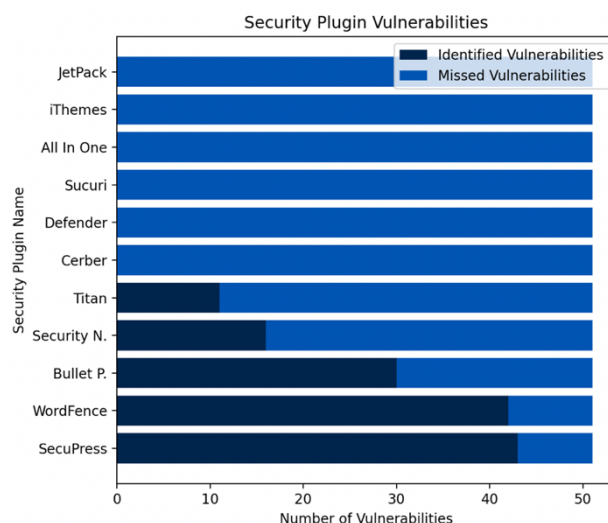


Figure 2.3 Efficacy comparison of popular WordPress security plugins (Murphy et. al., 2021).

The quantitative results shown in Figure 2.3, indicate that more than half (six out of eleven) of the security plugins, failed to detect any vulnerabilities. According to Murphy et al. (2021), security plugins namely, Cerber, Defender, All-in-One, iThemes, and JetPack requires users to upgrade to a premium version to access advanced capabilities that would detect more complex vulnerabilities (Murphy et al., 2021).

Considering that many of these free security plugins detect vulnerabilities using static analysis, it can be assumed that their signature databases deliberately lack vulnerability records from up-to-date sources such as the Exploit Database. Upgrading these plugins to premium, removes this restriction and enables the scanner to compare plugins with a wider range of records, enhancing its detection capabilities. From a user's perspective, the research indicates that popularity and low-cost are not reliable indicators of a plugin's efficacy, despite often relying on these factors when selecting plugins. Similarly concluded by Kasturi et al. (2022), WordPress website owners often rely on simple metrics such as plugin popularity, ratings, and reviews but unfortunately, these metrics do not provide a comprehensive or reliable measure of their efficacy (Kasturi et al., 2022). The fact that many of these security plugins perform poorly highlights a significant gap in security, leaving users vulnerable to potential threats such as delayed patches or suspended plugins.

2.2.2 Accessibility

Accessibility refers to the ease of use for users to access and manage WordPress security scanner plugins. Given that 29% of WordPress plugins with critical vulnerabilities did not receive a developer patch, the effectiveness of the assessed security plugins against this issue is unknown (PatchStack, 2022). Users must manually log in to the website's backend to check if they have installed a vulnerable plugin with or without an effective security scanner.

Despite the accessibility problem of security scanner plugins, Fragulis et al., 2018, highlights the advantage of popular CMSs like WordPress. Their advantages such as plugin administration and installation process does not require specialised programming knowledge, only administrative privileges, and a connection to the control panel. With a few clicks, users can download, activate, and administer plugins (Fragulis et al., 2018). However, there are also disadvantageous attributes to security scanner plugins, including resource usage. For example, many anti-virus plugins use task schedulers to run scheduled scans which may slow down the performance of the website especially if an event gets executed very frequently. Overall, the accessibility of security scanner plugins is two sided and is up to users to sacrifice high resource usage for security.

2.2.3 Compatibility

Compatibility refers to the ability of a security scanner plugin to operate with versions of the CMS or server. While some security scanner plugins may have specific server requirements, WordPress is known for maintaining backward compatibility and creating a developer-friendly environment (Williams et al., 2020). The system requirements for running a security scanner plugin are usually defined by the plugin developer and typically only require the latest version of WordPress to be installed.

Generally, WordPress does not require a powerful server or software, and it is commonly run on the LAMP stack. The current requirements stipulate that PHP version 7.4 or higher, MySQL 5.7 or higher, Apache, and HTTPS support are necessary to suitably run WordPress (WordPress, 2023). Overall, the compatibility of security scanner plugins is fairly low demanding of system requirements.

2.3 Impact of Security Patch Delays

Although this research was originally intended to focus on two specific demographic groups, it became apparent that in many cases the actual plugin users are not SMEs but rather a third-party entity responsible for the development and design on behalf of the business. Typically, organisations outsource these tasks to external marketing agencies (Stöver et al., 2023). This shift affects perspective as SMEs may lack technical expertise and are often unaware of potential security concerns, such as data breaches caused by plugin patch delays, for which they may not feel responsible. The following diagram, Figure 2.4 illustrates this new perspective.

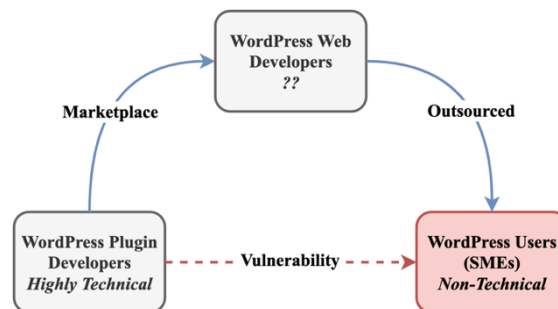


Figure 2.4 Visual representation of the research problem.

As shown in Figure 2.4, while SMEs are not responsible for developing plugin patches, they still bear the burden of costs associated with “*preventing, detecting, and correcting potential and actual breaches*” (Sen et al., 2020). Similarly, WordPress web developers are in charge of handling plugins, yet SMEs are the most vulnerable in this case. To better understand the impacts and effects of patch delays on users, the following section examines two case studies including the 2017 Equifax data breach and the 2022 WPGateway plugin exploit.

2.3.1 Case Study 1: Equifax Data Breach

The 2017 Equifax data breach is regarded as one of the largest cybercrimes in history in which approximately 147.9 million Americans, 15.2 million British citizens, and about 19,000 Canadian citizens had their private records compromised. The primary cause of the breach was attributed to Equifax's IT department, which delayed patching a known vulnerability in the Apache Struts server

software (Berghel, 2017). Equifax's oversight of the IT department resulted in a five-week delay in reporting the data loss to the public, despite the availability of a patch six months prior.

The hackers responsible for the breach had been actively searching for websites that had failed to update Struts, gaining access to Equifax's internal servers on May 12, 2017. The exploitation was used to steal internal credentials for Equifax employees. With this information, the hackers were able to search Equifax's credit monitoring databases under the guise of authorized users, performing more than 9,000 scans and extracting information into small temporary archives that were then transferred off the Equifax servers to avoid detection. The breach went undetected for 76 days until July 29, 2017, when Equifax discovered the intrusion. Equifax offered affected users settlement funds and free credit monitoring in a settlement with the United States Federal Trade Commission. However, the breach highlighted further faults in Equifax's system, including inadequate encryption and ineffective breach detection mechanisms, and insecure network design that lacked sufficient segmentation.

2.3.2 Case Study 2: WPGateway Zero-day Exploit

In 2022, the Wordfence Threat Intelligence team discovered an actively exploited zero-day vulnerability in the WPGateway plugin that affected more than 280,000 WordPress sites (Gall, 2022). The premium plugin linked to the WPGateway cloud service, provides users with a single dashboard for setting up and managing WordPress sites. The discovered vulnerability allowed unauthenticated attackers to insert an unauthenticated administrator. This administrator can effectively take over the entire site, leading to significant harm and loss for site owners.

The Wordfence team took immediate action upon discovering the vulnerability. They released a firewall rule to their premium, care, and response customers on the same day, September 8, 2022. The free version of Wordfence was set to receive the same protection 30 days later, on October 8, 2022. The team also reached out to the plugin vendor with their initial disclosure of the vulnerability on September 9, 2022. The vulnerability was identified as *CVE-2022-3180*, and certain details were withheld to prevent further exploitation. The Wordfence team urged all site owners who use the WPGateway plugin to remove it immediately until a patch was made available and to check for malicious administrator users in their WordPress dashboard.

2.4 Summary

Section 2.1 discussed two studies on WordPress plugin vulnerabilities. The first study, conducted by Mesa et al., (2018), aimed to identify the most frequent plugin vulnerabilities, the complexity of patching plugin vulnerabilities, and the relevance of plugin vulnerability-related discussions among developers. The study found that the most frequently encountered vulnerabilities between 2006 and 2016 were XSS, SQLi, and CSRF. Four evolution patterns for vulnerabilities caused by plugins were also identified, namely persistent, occasional, dormant, and prevalent. The study found that vulnerabilities caused by plugins tend to follow a persistent evolution pattern, with 76.31% of the vulnerabilities repeatedly reoccurring over the years. The replicated study showed that the most common WordPress plugin vulnerabilities encountered between 2006 and 2023 were XSS, SQLi, and CSRF. These findings were consistent with the results reported by Aborujilah et al. in 2022, who found that the most frequent vulnerabilities in web applications are XSS, SQLi, and CSRF. The study concluded that developers do not understand common vulnerabilities, which makes users vulnerable if plugin developers do not acknowledge the importance of secure programming.

Section 2.2 discussed the current-day solutions for reducing risks associated with WordPress websites. Current-day security scanner plugins were analysed in efficacy, accessibility, and compatibility. The empirical assessment conducted by Murphy et al. (2021) indicated that six of the eleven most popular and freely available plugins failed to detect any vulnerabilities. Moreover, many free security plugins use static analysis and lack vulnerability records from up-to-date sources, which limits their detection capabilities. Additionally, popular metrics such as plugin popularity, ratings, and reviews do not provide a comprehensive or reliable measure of a plugin's efficacy. These findings highlight a significant gap in security, leaving users vulnerable to potential threats. The accessibility of security scanner plugins was found to be two-sided. While the administration and installation process of plugins do not require specialized programming knowledge, users must manually check for vulnerable plugins. This is

important, given that 29% of WordPress plugins with critical vulnerabilities did not receive a developer patch.

Section 2.3 has discussed the impact of security patch delays on SMEs and third-party entities responsible for developing and designing plugins on behalf of businesses. SMEs may lack technical expertise and may not feel responsible for potential security concerns, such as data breaches caused by plugin patch delays, yet they bear the costs associated with preventing and correcting potential and actual breaches. Two case studies have been examined to understand the impacts and effects of patch delays on users. The 2017 Equifax data breach affected approximately 147.9 million Americans, 15.2 million British citizens, and about 19,000 Canadian citizens, and was attributed to Equifax's IT department delaying patching a known vulnerability in the Apache Struts server software. Secondly, the 2022, WPGateway Zero-day which exploit affected more than 280,000 WordPress sites, allowing unauthenticated attackers to insert an unauthenticated administrator and take over the entire site.

As a conclusion of the literature review, it is clear that SMEs often outsource the web development of their websites to third-party agencies (Stöver et al., 2023). While this means they may not be directly responsible for handling security issues, such as patch delays, this issue still falls under their legal obligation to ensure their website is secure. Unfortunately, many vulnerabilities still pose a threat to plugin users, including XSS, SQLi, and CSRF, which can impact SMEs who may not have the technical knowledge to handle these issues. Although security plugins can help mitigate some of these vulnerabilities, research suggests that over 55% of security plugins fail to detect them, leaving users vulnerable to attack. Furthermore, implementing these plugins can be resource-intensive, which can slow down a website. As a result, web developers may choose not to implement security plugins, leaving SMEs at risk of data breaches and other security incidents. The case studies reviewed in this literature highlight the impact of patch delays and the importance of addressing security vulnerabilities promptly.

In terms of project development, the literature indicates that safeguarding SMEs should be the primary concern given their inherent vulnerability compared to the other two parties involved. SMEs typically lack technical expertise, which impacts the research question two (Q2), indicating that either plugin or web developers may be responsible for making SMEs susceptible to threats. As such, the focus should be on protecting SMEs as a priority from XSS, SQLi and CSRF.

3. Methodology

The following section provides an in-depth overview of the research methodology to ensure consistency, validity, and transparency. It involves the descriptions of data collection, analysis, ethical considerations, and limitations. This section focuses on three distinct demographic groups: web developers, plugin developers, and WordPress users.

3.1 Web Developers

As conducted by Stöver et al. (2023), SMEs utilising WordPress often outsource their web development, giving developers control over how plugins are implemented (Stöver et al., 2023). Therefore, the aim of conducting primary research on web developers is to “*understand the actions of web developers towards their clients, specifically small to medium-sized enterprises, in relation to patch delays and suspensions.*”. For this reason, qualitative research design will be employed due to its flexibility, allowing for adjustments to be made as the project progresses. This will be in the form of a survey methodology to gather a diverse range of responses. The survey questions will be worded slightly differently, using the term 'your website' to refer to the website of web developers' clients. The following section, Table 3.1, provides a comprehensive description of the questions that will be asked from participants.

Table 3.1 Web developers survey questions.

No.	Question	What is the purpose?	What is the expected outcome?
1	How integral is your website to your business, and to what extent do website sales impact your business' cash flow?	To establish the background of the web developer. This can later be used to classify and analyse the responses thematically.	Ecommerce websites generate the majority of cash flow in an online business; however, this can depend on the number of visits.
2	What criteria do you typically use when selecting and installing plugins on your WordPress site? Do you base your decisions primarily on ratings, reviews, or order of appearance?	To gain insights into the decision-making process of web developers when selecting and installing plugins on their clients' WordPress sites, seeking to understand the criteria they use for making these choices.	Participants can provide examples such as rating, review or appearance based. Many of these answered are expected to be the same.
3	How do you stay informed about security updates and patches for the plugins you use on your WordPress site?	To get an idea of how web developers get security updates about their client's website.	It can be assumed that most responses will involve short descriptions such as blogs, security plugins or newsletter.
4	Do you utilise any type of anti-virus plugin on your WordPress website? If so, which one(s) do you use and why? If not, why have you chosen not to use any?	To explore the security measures implemented by web developers to minimise patch delays and suspensions. To compare the results, the study can refer to Murphy et al.'s (2021) research on the security plugins used by web developers.	Common responses to this question may include Sucuri or WordFence.

5	How often do you access the back end of your website to check for plugin updates or vulnerabilities?	To investigate how often do web developers check for vulnerabilities in their client's website.	Participants are likely to provide small responses such as monthly or yearly.
6	How comfortable do you feel with identifying and addressing security vulnerabilities in WordPress websites?	To further explore the level of awareness of vulnerabilities among web developers.	Expected outcomes of this question are comprehensive responses such as unawareness of XSS, SQLi or other vulnerability types.
7	Would you find it helpful to receive security notifications about plugin vulnerabilities on your WordPress website via email? Why or why not?	To understand the respondents' views on a potential solution that would safeguard WordPress users (SMEs) while also being beneficial for them.	The expected responses would outline the benefits of this approach in terms of improved website security and the ease of implementation for both developers and clients.

The duration of this survey will be two to three weeks, subject to reaching the intended minimum of twenty-five participant responses. This timeline is constrained by project deadlines, and therefore, efforts to contact freelance and agency developers will be limited to expedite the process. However, in general, twenty-five responses are sufficient for drawing assumptions and conclusions for a specific target audience. Sampling bias will be minimised by approaching a diverse range of sources including freelancers, and marketing agencies. To size bias further, developers will be selected from a wide range of countries including the United States, United Kingdom, and European Union. On the contrary, surveys may yield a large number of participants but with low-quality responses. Data analysis will involve both thematic and statistical analysis to identify common patterns and themes that can make conclusions. This process will ensure that the data is consistently, validly, and transparently represented, and relevant insights can be drawn.

3.2 Plugin Developers

According to the study conducted by Mesa et al. (2018), the most commonly encountered vulnerabilities in WordPress plugins are XSS, CSRF, and SQLi. Since these vulnerabilities fall into P1, it is probable that they will continue to persist in the years to come (Mesa et al., 2018). For this reason, the aim of conducting primary research on plugin developers is to “*understand the behaviour of developers and mitigation methods they use to avoid vulnerabilities caused by plugin patch delays and suspensions.*”. To conduct this research, qualitative research methodology will be employed because of its flexibility, allowing for adjustments to be made as the project progresses. The research will utilise surveys to gather a diverse range of responses. The data in Table 3.2 presents in-depth explanations of the survey questions that will be posed to the participants.

Table 3.2 Plugin developers survey questions.

No.	Question	Purpose	Outcome
1	How many years have you been developing plugins for?	To assess the skill set of the plugin developer, distinguishing between a professional and a hobbyist.	The response is expected to provide a rough estimate of the number of months the developer has spent developing plugins.
2	Do you receive payment for developing plugins?	To establish the background of the plugin developer, specifically whether plugin development is pursued professionally for profit.	The expected responses should provide details on whether the developer engages in full-time or part-time plugin development and the focus of their plugin development activity.
3	Have you ever experienced a delay in releasing patches or updates while developing WordPress plugins? If yes, what were the primary reasons for the delay?	To gather information about whether the plugin developer has faced any security issues during the plugin development process, which caused delays in releasing patches.	The expected outcome of this question is to receive detailed responses about the developer's experience and how they resolved the issue.
4	Have you ever had a plugin suspended by WordPress? If yes, what were the reasons behind the suspension?	To gather information regarding whether the plugin developer has experienced plugin suspension by WordPress and the underlying reasons for it.	The expected outcome of this question is to gain insights into the situation that led to the suspension and the steps taken to address it, if any.
5	What resources do you use to keep up to date with changes in the WordPress platform and other plugins that your own plugins interact with?	To obtain insights into the support documentation or text-based resources leveraged by plugin developers.	The expected responses should comprise detailed information and justification on the use of official documentation, community blogs, and newsletters.
6	How do you prioritise bug fixes and new feature development in your plugin?	To get an understanding of how developers manage security in plugin development. This means how they aim to avoid plugin patch delays and suspensions.	The expected response should include a detailed response about user feedback, prioritisation, and severity management.

7	How do you manage support requests and communicate with users of your plugin?	To identify potential communication issues between developers and users, which could be contributing to delays in patching and, ultimately, plugin suspensions.	Expected responses include discussions around social media, email supports or automated ticketing system.
8	What steps do you take to ensure the security and privacy of your plugin and its users?	To understand the steps taken by the developer to ensure security. This question sheds light on potential problems that could cause plugin patch delays and suspensions.	Expected responses should include a description of secure coding practices to prevent common vulnerabilities such as XSS and SQLi.

The survey is expected to last for two to three weeks, with the minimum target of fifteen to twenty responses. The timeline is constrained by project deadlines, which may limit the effort to contact freelance plugin developers to expedite the process. The goal is to ensure the survey is conducted within the required timeframe while obtaining a representative sample of the target audience. A potential advantage of surveys is their ability to be distributed widely among plugin developer forums, allowing for a large number of developers to be reached and potentially persuaded to complete the survey. However, one drawback of relying solely on surveys is that responses may not be as comprehensive or detailed as those obtained through interviews. In fact, finding enough qualified interviewees who are willing to devote time to the project is a major challenge due to developer's time, work, and commitments. Data analysis will involve both thematic and statistical analysis. Statistical analysis will be done by utilising the "matplotlib" Python module. Thematic analysis will be made via identifying common patterns and themes that can make conclusions. This process will ensure that the data is consistently, validly, and transparently represented, and relevant insights can be drawn.

3.3 WordPress Users

SMEs utilising WordPress are arguably the most important focus of the project. According to the literature review, SMEs using WordPress are the most vulnerable players in this issue due to their limited technical expertise and security awareness. The aim of conducting primary research on web developers is to identify their mistakes and determine ways to reduce their vulnerability to plugin patch delays and suspensions. To collect data, quantitative research methodology will be employed. This approach allows for easy distribution and yields a larger number of consistent responses. The research will employ two methodologies: a survey and a concept software.

Table 3.3 WordPress users survey questions.

No.	Question	Purpose	Outcome
1	How many employees does your business have?	To ensure that the research is targeted specifically towards SMEs, the initial question seeks to screen out responses that do not originate from small to medium-sized enterprises.	The expected response format is a single selection ranging from zero to five hundred employees.
2	Do you outsource your WordPress web development?	It may be the case that businesses may outsource their web development. In order to ensure the reliability and validity of assumptions, it is necessary to inquire about this practice from SMEs.	The question offers multiple choice options for the responder to select from, including "yes", "no", or "sometimes".

3	How would you describe your WordPress web development skills?	The literature review has revealed that there may be an incongruity in the technical skills of WordPress users. In order to ensure the credibility and validity of the assumptions made, it is imperative to seek input from subject matter experts (SMEs) on this matter.	The anticipated outcomes of this inquiry are graded levels of understanding and skill in technical language, ranging from low to high.
4	Have you ever experienced a security incident that has affected your WordPress website?	To determine whether SMEs have experienced security incidents that have impacted their WordPress websites.	The inquiry will be posed in the form of a multiple-choice question, with expected responses such as yes or no.
5	How do you communicate with your web developer regarding plugin updates and website security?	To investigate communication challenges between developers and users.	Responses will be in the form of multiple-choice options such as social media, WordPress forums, or email.

This survey is expected to run for one to three weeks, with the minimum target of twenty to twenty-five responses as the project is time constrained and therefore there is limited time to contact SMEs and request them to fill out the survey. Data analysis will involve both statistical analyses to identify common patterns and themes that can make conclusions. Statistical analysis will be done view the “matplotlib” Python module. This process will ensure that the data is consistently, validly, and transparently represented, and relevant insights can be drawn.

3.4 Tools and Requirements

The plugin will be developed using a local webserver consisting of Apache 2.4, MySQL 5.7, and PHP 7.4. Visual Studio Code will be used as the primary Integrated Development Environment (IDE) to facilitate the coding process.

3.4.1 Functional Requirements

Functional requirements are specific features and functions that a software solution must have to meet the needs of its users, while non-functional requirements are criteria that a software solution must meet to be considered successful, such as performance, security, or usability.

1. The software must scan through the user's currently active WordPress plugin directory.
2. The software must compare the version of the WordPress API to determine if a patch is available.
3. The software must send an automatic email notification to the user's outsourced web developer if an outdated plugin is detected.
4. The user must be able to select how frequently the scan is performed.

To illustrate the user's interaction with the proposed solution, Figure 3.1 depicts the software in a visual representation that includes both the front-end and back-end mechanisms. The three primary sections that the user can interact with in the front-end are Active Scan, Integrations, and Settings. The Settings tab is where the user will likely spend most of their time as they will need to enter the email address and name of the outsourced web developer.

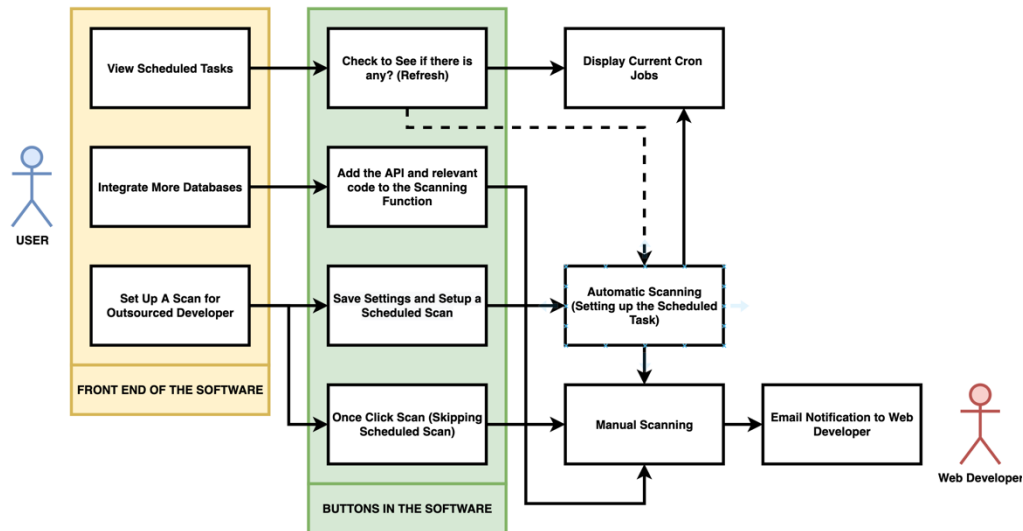


Figure 3.1 UML interaction diagram

3.4.2 Non-Functional Requirements

Non-functional requirements are constraints that specify how well the concept software performs its functions. Non-functional requirements are concerned with qualities such as performance, security, reliability, maintainability, usability, and scalability. The following four non-functional are outlined for the concept software.

1. **Performance:** The software must scan the user's WordPress plugin directory and compare the WordPress API version quickly and efficiently.
2. **Reliability:** The software must reliably detect outdated plugins and send an automatic email notification to the user's outsourced web developer.
3. **Maintainability:** The software must be easy to maintain and update to ensure that it continues to function correctly as WordPress updates and new vulnerabilities are discovered.
4. **Usability:** The software must be easy to use and understand, with clear instructions for installation and setup.

3.5 Concept Software

The following section presents the proposed software concept aimed at addressing the research question (as highlighted in Section 1.2). Developing a successful software solution entails various processes, including design, implementation, and testing, to ensure its dependability and efficacy.

3.5.1 Design

The solution is designed as a WordPress plugin that scans through the user's currently active WordPress plugin directory and compares the version of the WordPress API to determine if a patch is available. If an outdated plugin is detected, an automatic email notification will be sent to the user's outsourced web developer, informing them of the issue. This helps to ensure that website vulnerabilities are addressed promptly, even if the user lacks the technical expertise to identify and address such issues. The plugin is easy to install, and users can select how frequently the scan is performed, ranging from every minute to every month. The scan is set up via a task scheduler, which the developer can easily remove using the plugin dashboard. The task scheduler function is already inbuilt into the WordPress core, but it has been extended to offer this extra functionality. The following diagram, Figure 3.2, shows an overview of the concept software.

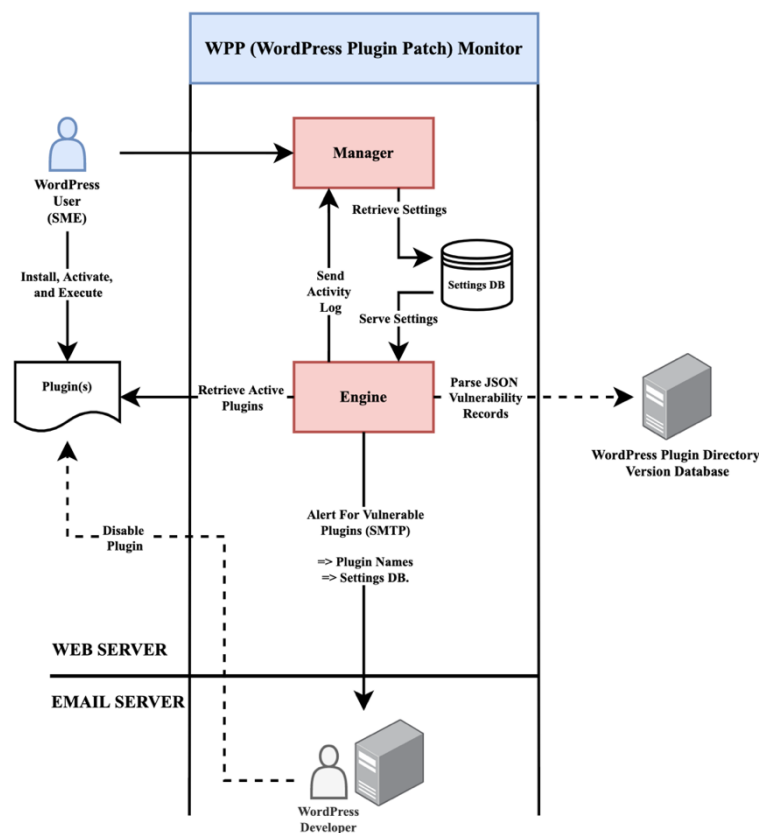


Figure 3.2 WPP Monitor Plugin functionality overview.

The development of the proposed WordPress plugin will follow a rigorous approach using the agile methodology for its benefits, including a flexible and iterative approach to development, regular feedback loops, and continuous integration and delivery. This approach will ensure that the plugin meets the requirements, is user-friendly, and is delivered within the expected timeframe.

3.5.2 Testbed

To collect data, a test will be conducted on WordPress websites that are hosted on various servers and have plugins installed randomly. The researcher in charge of this project already has four WordPress websites made dating back to 2021, set up online, which will be moved offline to a locally hosted environment for testing purposes. The following steps will be taken to accomplish this:

1. Installation of a backup-plugin on the online WordPress website.

2. Deployment of an offline (local) version of a WordPress website.
3. Restoration of the backed-up WordPress website from the online source.

Firstly, a premium version of the “All-In-One Migration Tool” plugin will be downloaded to backup and obtain the backup file. Secondly, a local environment will be established using either the “Local” software for Mac or XAMPP for Windows. The WordPress installation will be configured based on the official documentation and checked for any modifications, such as pre-installed plugins or pages, which will then be removed to ensure a clean WordPress version. Lastly, the “All-In-One Migration” plugin will be downloaded once again, and the backup file obtained from the online website will be restored. The WPP Monitor plugin, the implemented product from this project, will be installed, and the “one-click” functionality will be used to bypass cron jobs. The plugins that were flagged by the WPP Monitor plugin will then be checked.

The results of this study will be analysed using the ‘matplotlib’ Python module through a quantitative research design. This design is appropriate since the study will focus on quantifying the number of outdated plugins reported to a pre-set email address. Due to the identified vulnerabilities by Mesa et al. (2018), this conceptual software will not be released for public such as testing on real-world SMEs until adequate security measures are implemented.

3.6 Data Collection and Management

Before distributing surveys, participants will be informed about the purpose and nature of the research and asked to provide informed consent by ticking the mandatory fields on the Online Surveys platform. To ensure the participant’s confidentiality and anonymity in the process of collecting, processing, and storing data, password protection will be implemented, and data will be stored on OneDrive. In case a participant shares sensitive information, the unnecessary data will be disposed to comply with the following regulations:

1. Data Protection Act 2018 for the United Kingdom.
2. General Data Protection 2020 for the European Union.
3. California Consumer Privacy Act 2018 and New York Privacy Act 2021 for the United States.

The distribution process of the survey will involve posting the survey link on professional platforms and WordPress forums such as Stack Exchange, Stack Overflow or LinkedIn. The post will include the title of the research, the aims and objectives and information about eligible participants.

4. Implementation

The implementation chapter provides a comprehensive overview of the development process for the WPP Monitor plugin, the proposed automatic notification system software solution for monitoring WordPress plugin patches. The section follows the agile methodology to ensure efficient and timely completion of the project. The software comprises two critical components: the manager and engine. The subsequent section provides a detailed exposition of these components, offering a comprehensive understanding of the WPP Monitor plugin's inner workings. Code and survey results can be found at <https://github.coventry.ac.uk/6047CEM/10135169>

4.1 Manager

The manager component of the WPP Monitor plugin acts as the primary interface, enabling users to input, store and run the software based on their desired settings. This includes critical information such as the organization's name and email, as well as the outsourced developer's name and email. Users can also specify the frequency of scanning the site for available patches for the plugins. The manager comprises three key components, namely the “Active Scans”, “Settings”, and “Integrations” tabs.

The “Active Scans” tab displays all the task schedulers that have been set up, while the “Settings” tab allows users to customize their email address and preferences. Finally, the “Integrations” tab permits users to extend the capability of the software by adding more database, (refer to Section 8.1). The following code shows the inner mechanism of the settings tab in the manager.

```
<div class="tab-content">
    <?php
        // Include the appropriate PHP file depending on the current tab.
        switch($tab):
            case 'active-scans':
                include (plugin_dir_path( __FILE__ ) .
'manager/active_scans.php');
                break;
            case 'settings':
                include (plugin_dir_path( __FILE__ ) .
'manager/settings.php');
                break;
            case 'integration':
                include (plugin_dir_path( __FILE__ ) .
'manager/integration.php');
                break;
        endswitch;
    ?>
</div>
```

Listing 4.1 Dashboard code.

In Listing 4.1, the switch statement facilitates users in switching between the active scans, settings, and integrations tab. It is crucial to have code that is both efficient and easy to read, which can be extended or replicated by third parties. To achieve this, the functionalities have been split into separate files, and the "include" function is used to display the code within the switch statement. This enables seamless switching between different functionalities for the users. These files are being called from the manager folder which is located inside the templates folder. For more detail on the layout of the folders and files, the GitHub repository can be found in Section 1.

The "Active Scan" tab enables users to view the currently active task schedulers that have been set up through the plugin. These jobs are identified by the name "wpp_scan_event" and display information such as the frequency of execution in seconds (interval), the next scheduled run time, and any associated arguments including the names and emails of the web developer and user. The tab utilizes the

WordPress core functionality of the cron function, which generates an array of arguments that can be specified to call different items within the array. This feature allows users to view the settings of a website and assign outsourced web developers to specific tasks. In addition, the tab enables users to refresh the feed for newly set up task schedulers and remove unnecessary ones. The code for this functionality is displayed in Listing 4.2.

```

foreach ($filtered_crons as $timestamp => $cron) {
    foreach ($cron as $hook => $details) {
        //Filter any task schedulers under the specified name.
        if ($hook === 'wpp_scan_event') {
            echo '<tr>';
            echo '<td>' . $hook . '</td>';
            foreach ($details as $key => $value) {
                echo '<td>' . ucfirst($value['interval']) . ' (' .
ucfirst($value['schedule']) . ') . '</td>';
            }
            echo '<td>' . date('Y-m-d H:i:s', $timestamp) . '</td>';
            echo '<td>';
            foreach ($details as $key => $value) {
                foreach ($value['args'] as $argKey => $argValue) {
                    echo $argKey . ': ' . $argValue . ', ';
                }
            }
            echo '</td>';
            echo '<td><form method="post">';
            echo '<input type="hidden" name="wpp_delete_event_nonce"
value="" . wp_create_nonce('wpp_delete_event') . ">';
            echo '<input type="hidden" name="wpp_delete_event_timestamp"
value="" . $timestamp . ">';
            echo '<input type="hidden" name="wpp_delete_event_hook"
value="" . $hook . ">';
            echo '<input type="submit" value="Delete">';
            echo '</form></td>';
            echo '</tr>';
        }
    }
}

```

Listing 4.2 Active scan tab code.

The "Settings" tab aims to allow users to store and input their own data including their name and email, along with their web developers a name and email. This functionality is used by the email function that sends out the available patches to the specified developers. One issue that was encountered throughout the development of this functionality was the creating task the monthly cron jobs. This was caused by the WordPress core as it does not have this functionality without manually adding it to WordPress, it was therefore concluded that the best approach would be leaving out and alerting that users can only make use of this feature if it is added manually.

```
// Schedule the task scheduler based on the selected frequency
switch ($frequency) {
    case 'minute':
        wp_schedule_event(time(), 'minute', 'wpp_scan_event', array($name,
$email, $dev_name, $dev_email));
        break;
    case 'daily':
        wp_schedule_event(time(), 'daily', 'wpp_scan_event', array($name,
$email, $dev_name, $dev_email));
        break;
    case 'weekly':
        wp_schedule_event(time(), 'weekly', 'wpp_scan_event', array($name,
$email, $dev_name, $dev_email));
        break;
    case 'monthly':
        wp_schedule_event(time(), 'monthly', 'wpp_scan_event', array($name,
$email, $dev_name, $dev_email));
        break;
}
```

Listing 4.3 Settings tab code one.

The "Settings" tab provides the ability to save settings, including the creation of a new task scheduler. However, it also includes a feature to test the validity of an email address using a one-time button located next to the "save" button. This feature bypasses the setting of the task scheduler and proceeds directly to the scan and email functions. The primary goal of this software is to function correctly, and the inclusion of this testing functionality is therefore justified. The functionality operates by transmitting a POST request to either save or scan, depending on the desired outcome. Listing 4.4 displays the relevant code for this functionality.

```
// Save or Scan?
if (isset($_POST['save'])) {
    //Save the whatever is in input and create a task scheduler.
    save_function();
} elseif (isset($_POST['scan'])) {
    // Trigger the scan and email function immediately (skip cron).
    job();
}
```

Listing 4.4 Settings tab save or scan code.

The 'Integrations' tab is a future work feature designed to demonstrate the project's potential for extension. With additional databases incorporated into the software, users can specify specific vulnerabilities for comprehensive protection against XSS, SQLi, or CSRF. Currently, the code can save any length of string (API) in a database. Extending this functionality allows users to add the API to the scanning function and compare currently active plugins with WPScan vulnerabilities.

```

<form method="post" action="options.php">
    <?php settings_fields('data_storage_options'); ?>
    <?php do_settings_sections('data-storage'); ?>

    <!-- WPScan Database -->
    <h2>WPScan Database API</h2>
    <table class="form-table">
        <tr valign="top">
            <th scope="row">API Key</th>
            <td>
                <input type="text" name="api-key" value="<?php echo
esc_attr(get_option('api-key')); ?>" /><br>
                <p>To integrate with our forms you need an API Key. <a
target="_blank" href="https://wpscan.com/"><b>Generate</b></a></p><br>
                <div bis_skin_checked="1"><button data-
action="api_key_validate" data-nonce="4bfbdl4751" class="button"
id="api_key_button">Validate API Key</button></div>
            </td>
        </tr>
    </table>
    <hr>
    <?php submit_button(); ?>
</form>

```

Listing 4.5 Integrations tab code.

4.2 Engine

The WPP Monitor plugin's manager component is the core of the proposed software, consisting of two modules: the scanner and the mailer. The scanner module is responsible for ensuring that all plugins are up to date. The following diagram, Listing 4.6, shows the algorithm.

```

foreach ($plugins as $plugin_path => $plugin) {
    $plugin_data = get_plugin_data(WP_PLUGIN_DIR . '/' . $plugin_path);
    $plugin_version = $plugin_data['Version']; // Get plugin version
    $response =
wp_remote_get('https://api.wordpress.org/plugins/info/1.0/' .
$plugin_data['Name'] . '.json');
    if (!is_wp_error($response)) {
        $body = wp_remote_retrieve_body($response); // Retrieve response
        $plugin_info = json_decode($body); // Parse JSON data
        if (version_compare($plugin_version, $plugin_info->version, '<'))
        {
            $outdated_plugins[] = array(
                'name' => $plugin_data['Name'],
                'version' => $plugin_version,
                'disable_link' =>
wp_nonce_url(admin_url('plugins.php?action=deactivate&plugin=' .
$plugin_path), 'deactivate-plugin-' . $plugin_path)
            );
        }
    }
}
return $outdated_plugins;

```

Listing 4.6 Scan function code.

To do this, the scanner loops through all currently installed plugins and compares their version numbers with the latest version numbers listed on the WordPress API. The information obtained is then stored in an array called "outdated_plugins," which also contains a disabling link created by the algorithm. After this, the scanner compares the version number of each plugin with the latest version available on the WordPress API. If the plugin version number is lower than the latest version, the plugin is considered outdated, and its name, version number, and disabling link are added to the "outdated_plugins" array.

The second component, the mailer, is responsible for sending emails to the web developer when outdated plugins are detected. To achieve this, the mailer function makes use of an existing WordPress function that efficiently sends out emails using four attributes: the developer's email, subject, message, and headers. The headers are particularly important because it ensures that emails are sent out in a format that is easily readable to the recipient.

The email is sent out via SMTP, which is a protocol for sending emails over the internet. The headers contain information such as the sender's email address, the recipient's email address, and the subject of the email. To illustrate how the email function working, Listing 4.7 shows part of the email function code.

```
// Send the email and display a message
if(mail($dev_email, $subject, $message, $headers, $outdated_plugins)){
    // Show a success message to the user
    echo '<div class="notice notice-success is-dismissible"><p>Email sent
to ' . $dev_email . '</p></div>';
} else {
    // Show a success message to the user
    echo '<div class="notice notice-error is-dismissible"><p>Email failed
to send to ' . $dev_email . '</p></div>';
}
```

Listing 4.7 Email function code.

Figure 4.1 displays the settings tab of the proposed solution, providing a demonstration of its functionality through the accompanying screenshot.

Figure 4.1 Example of the agile development testing phase

5. Results and Analysis

The following section critically evaluates the research results of Chapter 3, to identify common themes and patterns. Question three and six in the Web Developers survey will be analysed statistically as many of the responses were found to be fitting numerical conclusions.

5.1 Web Developers Survey

The subsequent section critically evaluates the answers given by participants to the survey questions of Table 3.1. The first question results indicate that the importance of a website varies depending on business' nature. For example, one respondent said *'It's essential. It generates lots of revenue, as it makes me promote my business easily to my end users.'*. This suggests that for some businesses, such as e-commerce, a website can be critical to their success and generate significant cash flow. However, for other businesses, a website may only serve as an online presence and generate minimal revenue. For example, a respondent mentioned that *"I use my website mainly to showcase my products and services"*. In conclusion, for businesses that rely on their websites to generate revenue, such as e-commerce, maintaining their website is critical, as any downtime can result in loss of revenue.

The second survey question has shown several factors that come into play when utilising plugins from the WordPress plugin repository. The most commonly discussed topics are ratings and reviews, with many respondents indicating that the opinions of others who have used the same plugins are heavily relied on. For example, one respondent states, *"I typically focus on ratings and reviews when selecting and installing plugins on a WordPress site."*. Another says, *"I tend to research plugins and base it on reviews and ratings carefully."*. Other factors influencing plugin selection include functionality, appearance, and update history. For instance, one respondent notes, *"Often, I use first on the list and then check out their reviews and last update."*. Others emphasise the importance of security and compatibility with their sites. As one person puts it, *"I look up some basic and insightful advantages of plugins before I use them, like compatibility, updates, supports, and documentation."*.

The third question of the qualitative survey has revealed that web developers have several strategies for staying informed about security updates and patches for plugins used on their clients' WordPress websites. Some of the common methods mentioned include installing security plugins like Wordfence or iThemes Security, enabling automatic updates, subscribing to WordPress security newsletters, and following WordPress security blogs. For instance, one respondent stated, *"I install Wordfence or iThemes Security, enable automatic updates, subscribe to WordPress security newsletter, follow WordPress security blogs, and monitor WordPress support forums."*. However, not all respondents prioritize staying informed about security updates and patches for plugins. Some mentioned that they do not find it essential, and they only update plugins when necessary. For example, one respondent said, *"I do not because I don't know how to stay informed."*.

The fourth question responses can be grouped into three main themes. Firstly, some respondents use anti-virus plugins such as Wordfence, iThemes Security, Jetpack, WPScan, All in One WP Security & Firewall, and MalCare Security. These respondents use these plugins for a variety of reasons such as its ease of use, free package containing amazing features, real-time threat defense feed and blocklist to prevent access to known malicious IPs, being built by WordPress, its security, and its efficiency in doing the job they need it to do. For instance, one respondent said, *"I use Wordfence because of its firewall, malware scanner, security hardening, and real-time threat defense feed"*. Secondly, some respondents do not use any anti-virus plugin on their WordPress website because some do not feel the need to as there is caution exercised when it comes to making any update on their website. For example, one respondent said, *"No I do not. I only exercise caution when it comes to making any update on my website."*

Based on the responses to the fifth question, it was found that the majority of respondents (44%) check their website's backend at least once a week, while 20% check it on a daily basis. Additionally, 12% of respondents reported checking once every two weeks, and 8% check every two days. Only a small percentage of respondents (4%) rarely check the backend, while 16% check weekly. This data indicates that a considerable number of respondents (64%) check their website's backend frequently, indicating that the developer has a high priority for website security. From this, it can be concluded that web

developers are aware of website security however it can be assumed that it is the lousy mistakes that make their clients (SMEs) vulnerable to patch delays and suspensions.

For question sixth, the responses to the question of how comfortable individuals feel with identifying and addressing security vulnerabilities in WordPress websites are varied. Some respondents feel confident and knowledgeable about security concepts such as SQLi and XSS. For instance, one respondent said, *"I am very familiar with MySQL, SQL injection and even cross-site-scripting XSS ... I am very comfortable with it; it expands my knowledge."* Some respondents depend on tools such as WPScan to identify security issues, and some have precautions to avoid these types of attacks. While some respondents feel happy, others are not comfortable at all, and they rely on hosting providers for assistance. One respondent noted, *"I don't feel uncomfortable about it. Because the last time I had to contact my hosting provider to help with it."*

For question seventh, out of the 25 respondents, 20 indicated that they would find it helpful to receive security notifications about plugin vulnerabilities on their WordPress website via email. Respondents mentioned that email notifications would be faster, more accessible, and easier to reach. Some also noted that they frequently check their email and prefer to receive notifications that way. One respondent stated, *"I visit my email more often, so getting notifications about plugin vulnerability on my WordPress website would be the best option."* Another respondent mentioned that they *"Yes, I'd like it through email, As I will be updated ASAP, and I need to update the plugins to prevent attack"*. In contrast, five respondents stated that they would not find it helpful to receive such notifications via email, citing concerns about receiving too many emails or not using WordPress for high-class projects.

5.2 Plugin Developers Survey

The subsequent section critically evaluates the answers given by participants to the survey questions of Table 3.2. From question one, a range of experience levels when it comes to developing plugins for WordPress were found. The majority of respondents have been developing plugins for over two years, with some having over four years of experience. One individual even mentioned having six years of experience. According to one of the respondents, *"During the course of my career, I have been building and customizing plugins to achieve the required functionality."* This statement suggests that developers who develop plugins for WordPress are likely to have a background in software development and are skilled in customization and problem-solving. Overall, it can be concluded that the average number of years that developers have been involved in plugin development is two and therefore these individuals possess a significant amount of programming experience and understand the mechanisms behind plugins.

From question two, many respondents' answers vary, with some charging for every type of work on an hourly basis, others receiving payment always or sometimes, and some only earning through the premium version of their plugin or when users subscribe. Some also offer their plugins for free, while charging a fee for others. One respondent stated, *"Yes, because I am a programmer who develops plugins for various platforms and I always receive payment for my completed works,"* emphasizing the importance of payment for completed work as a developer. Another respondent mentioned, *"If my user goes for the paid version of my plugin, I earn through that."* highlighting the significance of offering premium versions to earn payment.

From question three, results have shown that delays can occur due to a variety of factors including technical complexities related to plugin code, compatibility issues with other plugins or WordPress versions, and server configurations. As one respondent noted, *"there are some external factors that do cause delays, such as unforeseen bugs, security breaches, or unexpected events that require my development team's attention."* Overall, delays in releasing patches or updates can be caused by a range of factors, both internal and external, which need to be addressed by plugin developers to ensure timely and effective updates.

In question four, participants were asked if they had ever had a plugin suspended by WordPress, and if so, what the reasons behind the suspension were. Out of the 17 responses, the majority (10) said they had not experienced a plugin suspension. The remaining participants gave a range of reasons for the suspension, including security vulnerabilities (2), violations of guidelines (1), poor user experience (1),

spamming and fake clicks (1), malicious code and malware (1), vulnerable to a virus (1), infringement on copyright (1), and outdated and incompatible with the latest version of WordPress (1). One participant responded that they had not personally had a plugin suspended, but it was possible as WordPress has its own terms and conditions. The collected responses indicate that although plugin suspensions are infrequent, they remain a significant challenge when they do occur. Plugin developers still make mistakes that result in suspensions, which in turn leaves end users vulnerable.

From question five, the most commonly mentioned resources were official WordPress documentation and blogs, social media groups, online communities, and forums. In addition, some respondents reported using third-party services that offer automated testing and monitoring of plugins while others mentioned industry blogs such as WP Tavern, WPMU DEV, and Torque. As one respondent put it, they keep up to date *“through Plugin documentation, blogposts and some Community forums where the latest news are being shared.”*. While some respondents were aware of the security issues concerning not using official documentation, most users were using third-party websites to stay up to date with security updates.

For question six, when prioritising bug fixes and new feature development in their plugin, developers consider several factors. These include the severity of the issue, impact on users, user feedback, development effort, and business goals. Respondents prioritise critical bugs that affect the functionality of the plugin and have a significant impact on users' experience. They also pay attention to user feedback and prioritize requests that have been requested by a significant number of users. As one respondent put it, *“Based on my user feedback and experience, if my plugin is slowing down a website or causing errors, this can negatively impact my user experience, and therefore I should give it a high priority for fixing.”*.

For question seven, the majority of respondents use a ticketing system or a dedicated support email to manage support requests and communicate with users of their plugin. Quick and courteous responses are emphasized, along with clear and concise instructions to resolve issues. Many also maintain a knowledge base to help users self-serve and provide regular updates on the status of requests. Some respondents also use social media and a dedicated email for support requests. One respondent state, *“I have an email for that, where end users send reports,”*. Overall, the responses demonstrate a variety of approaches to managing support requests and communicating with users, with a focus on responsiveness and helpfulness.

For question eight, the majority of the sixteen respondents highlighted the importance of regular updates and code reviews to identify and fix vulnerabilities. Some also mentioned using secure coding practices, such as avoiding common vulnerabilities like SQLi and XSS vulnerabilities. Overall, the respondents' approaches to ensuring security and privacy in their plugins involve a combination of technical measures, regular updates, and user education. As one respondent put it, *“Usually I target best ever practices, and not to use shortcuts, like using secure coding practices, through this, I'm sure users are secured.”*.

5.3 WordPress Users Survey

The subsequent section critically evaluates the answers given by participants to the survey questions of Table 3.3. The WordPress survey aimed at businesses was conducted over a period of six days from March 10th to March 16th. Due to reaching the intended sample size of 25 responses, the survey was stopped early. The survey's first question has revealed that two respondents (8%) claimed to have between 201 and 500 employees, rendering them ineligible for the survey. Consequently, they have not been considered and therefore this could improve the survey's validity, but it may suffer from sampling bias and could be less representative of SMEs in the WordPress community.

Based on the survey results, 39.1% of participants represented small businesses with one to ten employees, 34.8% represented small businesses with eleven to fifty employees, and 26.1% represented businesses with 51 to 200 employees. Figure 5.1 presents the distribution of employee counts for survey participants.

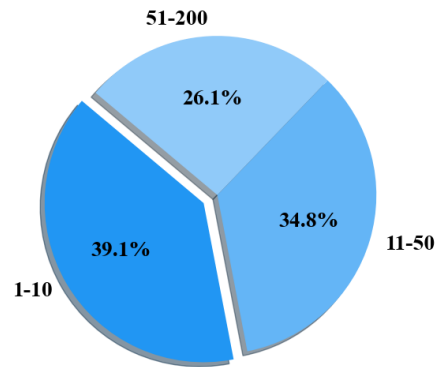


Figure 5.1 Pie chart: How many employees does your business have?

In the survey's second question, it was revealed that over 52.2% of SMEs opt to outsource their web development, while 26.1% sometimes choose to do so. Only 21.7% of SMEs do not outsource their web development. These findings further support the argument put forth by Stöver et al. (2023), who discovered that many businesses opt to outsource their web development or marketing needs to third-party agencies. As a result of this research, it is evident that SMEs also fall into this category. Figure 5.2 presents the distribution of outsourced WordPress web development.

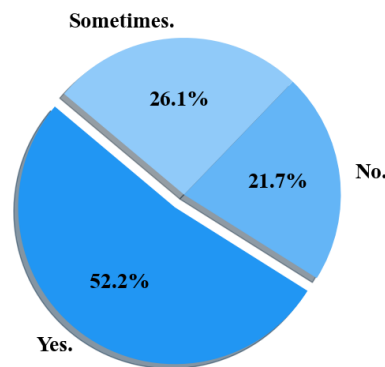


Figure 5.2 Pie chart: Do you outsource your WordPress web development?

Survey question three reveals that only 21.7% of SMEs possess advanced WordPress web development abilities, such as programming or implementing specific functionalities. Approximately 34.8% of participants comprehend WordPress in terms of basic updates, while 43.5% possess an intermediate level of understanding. These findings suggest that the majority of WordPress users lack advanced skill sets, including programming, and may not be aware of potential security threats such as XSS, CSRF, and SQLi. Figure 5.3 illustrates the distribution of WordPress development skills. Advanced refers to possessing advanced programming skills, intermediate refers to having some programming skills but not being highly technical, and basic refers to understanding how to install plugins.

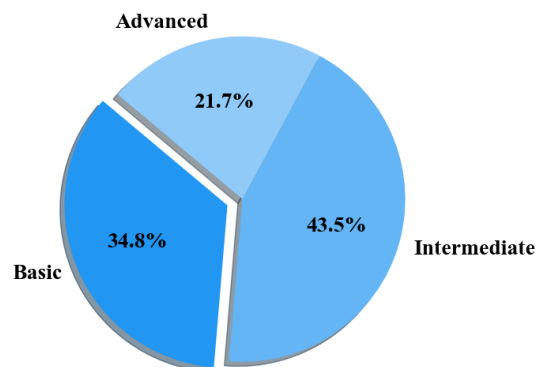


Figure 5.3 Pie chart: How would you describe your WordPress web development skills?

According to survey question four, 60.9% of SMEs experienced a security incident that impacted their WordPress website, with 8.7% unsure if they had. Only 30.4% of respondents reported no security incidents. These results indicate that SMEs face security issues with their websites, leading them to seek assistance from developers due to their limited knowledge on security. Figure 5.4 illustrates the breakdown of security incidents affecting SME websites.

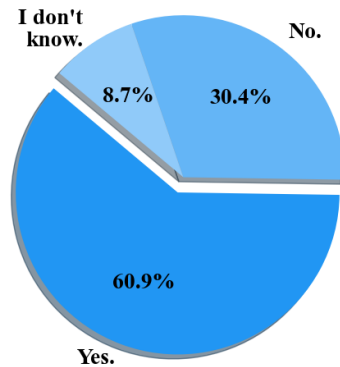


Figure 5.4 Pie chart: Have you ever experienced a security incident that has affected your website?

According to question five of the survey, 39.1% of SMEs reported having regular communication with their web developer for handling all plugin updates. The decision on which plugins to prioritize is mutually agreed upon by the developer and the SME. Similarly, another 39.1% of SMEs reported that their web developers handle all plugin updates and website security without requiring any input from the SME. Surprisingly, 21.7% of SMEs only reach out to web developers regarding plugin updates or security issues when a problem occurs. Figure 5.5 displays the distribution of communication levels between SMEs and outsourced web developers.

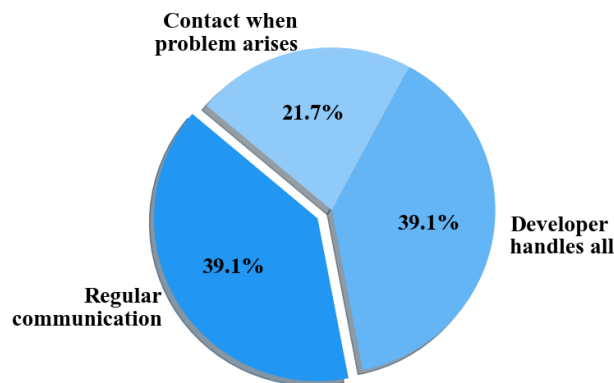


Figure 5.5 Pie chart: How do you communicate with your developer regarding website security?

5.4 Concept Software

According to the results, all of the plugins tested, except for Elementor Pro Version 3.10.3, were found to have no known vulnerabilities at the time of testing. It is unclear from the information provided why the notification for Elementor Pro was unsuccessful. Table 5.1 shows the successful detection of outdated vulnerabilities of the four websites.

Table 5.1 The concept software's efficacy.

No.	Backup Name	Outdated Plugin and Version	Notification
1	Backup #1736	Elementor Version 3.10.2	Successful
		WooCommerce Version 7.3.0	Successful
		Carousel Slider Block for Gutenberg Version 1.0.6	Successful
		Contact Form 7 Version 5.7	Successful
		Elementor Pro Version 3.10.3	Unsuccessful
		Pods – Custom Content Types and Fields Version 2.9.11.1	Successful
2	Backup #6532	MailPoet Version 4.8.1	Successful
		Elementor Pro Version 3.10.2	Unsuccessful
		Grids Version 1.3.2	Successful
3	Backup #1341	WPify Woo Czech Version 3.8.4	Successful
		TuriTop Version 1.0.5	Successful
		Elementor Pro Version 3.9.1	Unsuccessful
4	Backup #5353	Lock Login Version 0.1.5	Successful
		Elementor Pro Version 3.11	Unsuccessful
		BuddyPress Docs 2.1.3	Successful

6. Discussion

According to the findings of the analysed research, over 52% of SMEs outsource web development to third-party web developers, while 26.1% of SMEs occasionally outsource. This outsourcing trend is likely due to the complexities of web development, as over 34% of SMEs reported having basic-level skill sets in this area. The major problem is that many web developers make mistakes by leaving SMEs vulnerable to patch delays and plugin suspensions. One common mistake is selecting plugins based solely on ratings and reviews, even though this metric has been found to be unreliable by previous researchers such as Kasturi et al. (2022). Another mistake is installing security plugins that require users to upgrade to a premium version to access advanced capabilities, such as those found in Cerber, Defender, All-in-One, iThemes, and JetPack, similarly identified by Murphy et al. (2021). It is presumed that these mistakes are related, given that 60.9% of SMEs have reported being impacted by a cyber-attack that affected their website. Despite these mistakes, web developers are generally aware of vulnerabilities such as XSS and SQLi, and most check the backend of their sites at least once a week. Therefore, while web developers cannot be solely blamed for SMEs' vulnerability to patch delays and suspensions, plugin developers also make mistakes that contribute to these issues.

Based on the survey results regarding the WordPress plugin development, it is evident that even with a median experience of over two years, the release of updates can still be delayed due to a variety of internal and external factors. These factors include technical complexities, compatibility issues, and unforeseen events that developers must address. Plugin suspensions may also occur due to several reasons, including guideline violations, security vulnerabilities, poor user experience, outdated compatibility, and containing malicious code. It is essential for developers to take these factors into account when creating and updating their plugins to ensure the continued functionality and security of their products. On the other hand, plugin developers are reported to use resources for staying up to date with WordPress development, such as official documentation and blogs, social media groups, online communities, and forums. Developers prioritise bug fixes and new feature development based on severity, impact on users, user feedback, development effort, and business goals, using a variety of approaches to manage support requests and communicate with users. Security and privacy are ensured through a combination of technical measures, regular updates, and user education, including using secure coding practices, avoiding common vulnerabilities like SQLi and XSS vulnerabilities, and following best practices.

Despite the identification of the causes, reducing the risk of vulnerability resulting from plugin patch delays and more explicitly suspensions remain challenging. Nevertheless, after testing the conceptual software, it can be inferred that setting up a notification system for SMEs to alert their outsourced web developers can substantially mitigate most risks associated with becoming vulnerable due to patch delays. As previously analysed, many developers appreciate receiving security-related notifications, which can safeguard both SMEs and web developers from the negative consequences of patch delays. The results have shown that the software detects at least 83.3% of available patches. Even with the availability of multiple anti-virus software options, their complex and resource-intensive nature can potentially hamper the performance of websites belonging to SMEs. Therefore, incorporating the aforementioned concept software can prove advantageous. Nonetheless, it should be noted that unlike WordFence, premium plugins such as "Elementor Pro" cannot be brought to the attention of users due to the absence of such features in the WordPress API.

7. Project Management

The subsequent section outlines the project management methodologies employed, including the utilisation of the agile software lifecycle model, an updated Gantt chart and key changes made throughout the project while also exhibiting the legal and ethical implications of the research. Finally, the section includes a reflection on the supervision process, highlighting the student's experiences and the impact on the overall research process. By addressing these essential aspects, this section ensures the credibility and reliability of the research.

7.1 Agile Software Methodology

Throughout the implementation process of the concept software, the agile software development methodology was used to ensure the successful creation of the WordPress plugin. This methodology enabled the project to be completed in iterative stages, with regular feedback and adjustments made along the way by the supervisor. By breaking down the project into smaller, manageable tasks and prioritizing them based on the SMEs needs, the development process was more efficient. The following diagram, Figure 7.1 shows the agile software methodology.

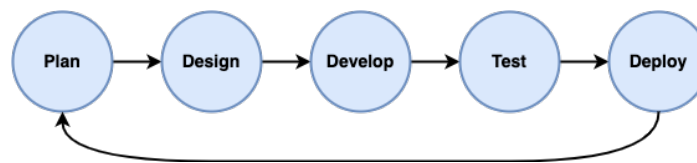


Figure 7.1 Overview of the agile software development cycle stages

7.2 Legal, Ethical and Social Constraints

Legal constraints encompass the regulations and policies that have limited the research's scope and implementation. As the data methodology involved collecting data from participants via an online survey from the European Union, the General Data Protection Regulation 2018 had to be implemented. This involved ensuring the participants' confidentiality and anonymity in collecting, processing, and storing data involved some sort of password protection. Similarly, data was also collected from the United Kingdom, so the research had to comply with the Data Protection Act 2018. This meant that before collecting any data, participants were informed about what the data was for to protect the privacy of the individuals whose data was collected. The California Consumer Privacy Act 2018 and the New York Privacy Act 2021 was also implemented as some individuals were from the United States. This meant that the research had to provide individuals with the right to know what personal information is being collected about them, the right to request the deletion of their personal information and the right to opt out.

Considering the legality of extending the WordPress framework and working with its source code, the GNU General Public License, version 2 (GPLv2), was considered. The license terms were reviewed carefully to ensure compliance and avoid violating the conditions set forth in the license by designing the plugin in a manner that is compatible with the terms of the GPLv2.

Ethical constraints, relate to the ethical principles and values that guide professional conduct and research activities. For example, as the project involved conducting qualitative research through online surveys with WordPress web developers, plugin developers, and users, the ethics board deemed the research to pose a medium risk. As a result, it was necessary to obtain ethical approval to safeguard the researcher and the university from any harm or legal issues that might have arisen. The Professional Code of Ethical Practice of the British Computer Society was used to provide guidelines for ethical research conduct. After discovering the diverse range of vulnerabilities in the literature review, the decision was made not to distribute the concept software and test it on SMEs due to the lack of time for utilizing effective secure programming practices. If the plugin had been distributed and installed, it could have put SMEs at risk of having their website compromised. Instead, it was decided that testing it on a local host would be more practical, given the importance of ethics.

Social constraints refer to the contextual factors surrounding this research, such as cultural and political backgrounds. It is for this reason that throughout the surveys data collection, a wide variety of

demographic groups were invited to participate in the questionnaire to reduce sampling and population bias. As a consequence of this, the research may become more representative of the broader WordPress community which includes SMEs from around the United States (California), European Union and the United Kingdom. The project was also following legal constraints and as a consequence it did not violate any social interactions which could have negatively impacted individuals' expressions of thoughts and feelings.

7.3 Student Reflection

The literature review presented several challenges that required careful thought. Firstly, sources needed to be more extensive in finding appropriate literature related to WordPress plugin development. To overcome this challenge, I requested more time from my personal supervisor and expanded my search to include sources about software development as a whole, utilising multiple source databases, such as IEEE Xplore, Science Direct, and ACM Digital Library. Another challenge was the limited availability of up-to-date research on the topic, with an average of three-year-old studies being the norm. To address this challenge, I consulted with my supervisor on the suitability of these research papers and repeated the methodology of several earlier studies, such as Mesa et al.'s (2018) study on plugin vulnerabilities, to compare their findings with the current state of plugin development. This approach allowed me to obtain a more comprehensive understanding of the issue, despite the need for more recent research.

The methodology chapter introduced various challenges that required me to rethink the original primary research plan from the proposal. Firstly, one-to-one interviews were removed as once I reached out to web developer agencies and freelancers, they did not want to participate, presumably since of the developer's time, work, and commitments. However, I overcame this by rethinking the process and discussing the problem with my supervisor. As a result of the meetings, it was clear for me to focus on doing a survey that can effectively collect results that do not hold high population and sampling bias. Overall, I was satisfied with the in-depth responses; however, if I redo this research, I will most likely keep an interview by paying for their time or giving them something in exchange. However, this would have to be reported to ethics approval, which could limit the research.

During the concept software implementation several challenges arose. One of the main challenges involved the managing a large number of items listed within the scheduler array. To overcome this issue, the agile methodology was implemented, and tests were performed throughout the development phase to ensure the functionality and effectiveness of the code. Eventually, the challenge was successfully resolved. Another issue arose during the deletion of scheduled tasks using AJAX. This issue was subsequently resolved by altering the functionality to a POST function that removed the necessary jobs. The implementation of agile methodology allowed for efficient and high-quality progression of the program. In terms of aesthetics and user experience, the implementation of HTML tables is highly appropriate. A well-designed user interface is crucial for a positive user experience, and using HTML tables can help achieve this goal. Additionally, code efficiency was taken into consideration, and to ensure optimal efficiency, the function was encapsulated within a for loop.

Overall, the project has provided me with valuable insights into the multifaceted nature of research. Although I was ill between week six and week seven, I managed to stand up and catchup with the workload relatively well. I have gained significant experience and a better understanding of the research process. If I were to undertake a similar project in the future, I would approach it differently by dedicating more time to reading and analysing the relevant material.

7.4 Initial Gant Chart

The initial Gantt chart before the project began is depicted in the following diagram. However, the first three weeks, which involved developing the proposal, were not included in this chart. At that stage, it was uncertain how the project would be completed, and external factors such as illness or personal issues were not considered. As a result, the following Gantt Chart, Figure 7.2, was revised (refer to Figure 7.3).

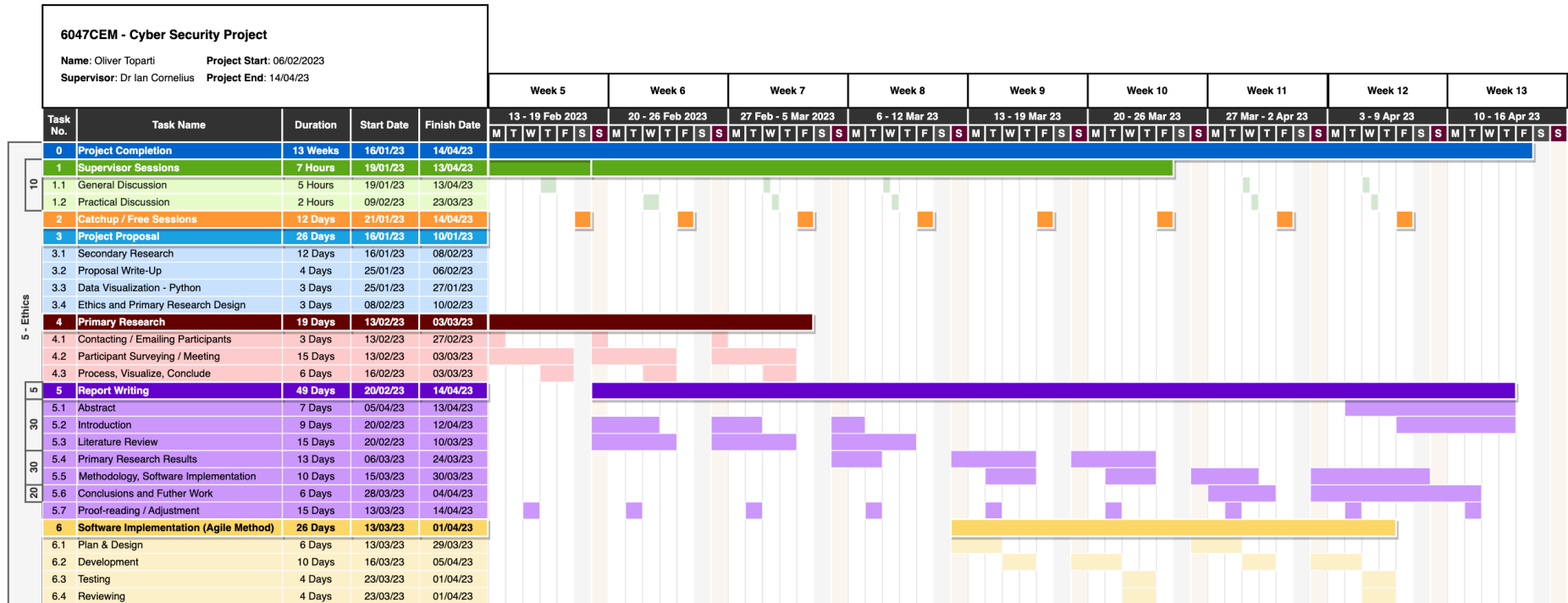


Figure 7.2 Initial Gantt chart of project tasks and milestones.

7.5 Post Gantt Chart

As seen on Figure 7.3, due to the researcher's illness, the project timeline was extended by one week. The university approved the extension after receiving appropriate medical documentation, and the response time was 48 hours. During the extension, the researcher focused on recovering and organizing their work while eliminating some tasks, such as report writing in LaTeX and more in-depth survey analysis, while maintaining other tasks due to a quick recovery. Figure 7.3 shows that the initial week 13 activities were rescheduled to week 14. Additionally, workload from other modules, such as Exploit Development, caused delays in meeting deadlines, resulting in the cancellation of supervisor sessions scheduled for weeks 10 and 11. Limited progress was made on the project during this time, particularly in the development of the concept software.

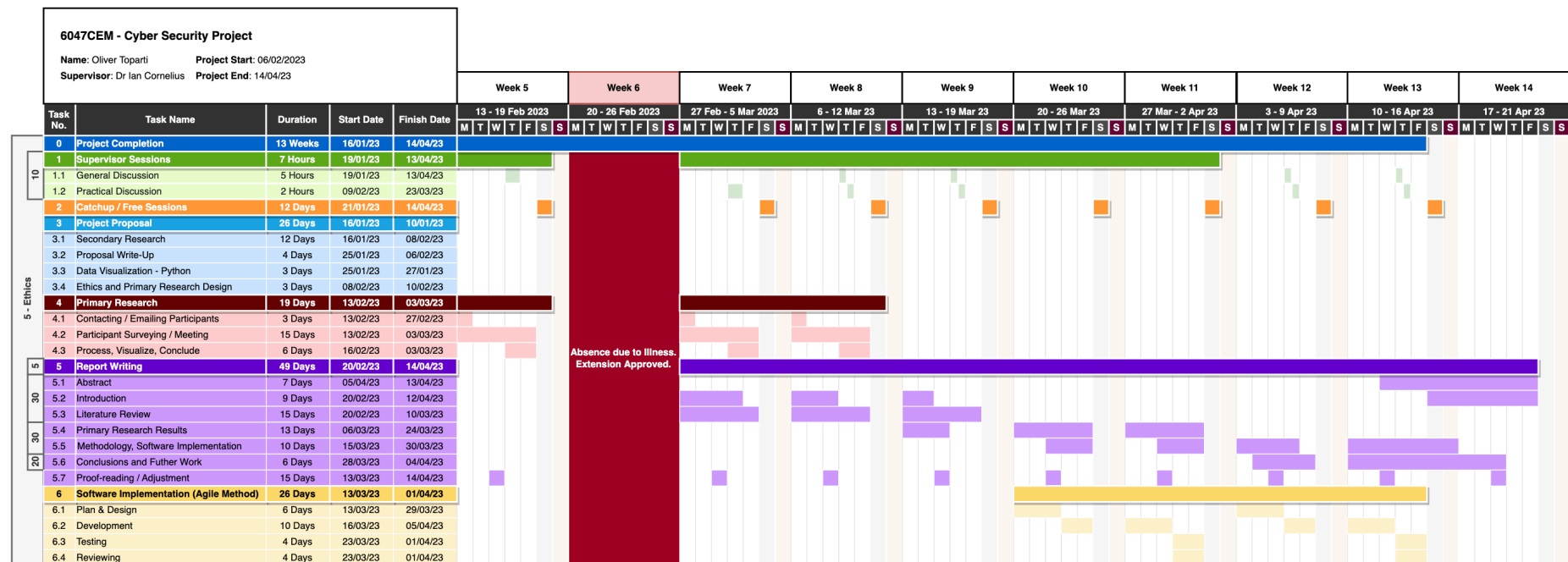


Figure 7.3 Post Gantt chart of project tasks and milestones.

8. Conclusion

The following section concludes the report by summarising the initial aims and objectives, the findings of the project, whether the research questions have been answered, and outlining potential future work.

The aim of this research was to answer, “*What are the causes and how can users reduce the risks of becoming vulnerable from WordPress plugin patch delays and suspensions?*”. This question was addressed by the conducting secondary research on the frequency, complexity, and relevance of plugin vulnerabilities. Furthermore, there was analysis on the current solutions in terms of efficacy, accessibility, and compatibility and presented two case studies to understand their impact. Through the literature review, it was discovered that the cause of risks stemming from plugin patch delays and suspensions is not the responsibility of the SME itself but rather that of the outsourced web developer. This statement is further supported by the qualitative survey of web developers described in Section 6. To measure the success of the project, it is essential to reflect and answer the question of Section 1.2.

In order to address RQ1 and RQ2, it is necessary to take into account the contributions of all users, plugin developers, and web developers to patch delays and suspensions. Firstly, it should be noted that 52% of SMEs outsource web development, meaning that the control over website management is generally outside their purview, as 31.1% of web developers handle all development, including plugin installation and management. Secondly, web developers are responsible for installing security plugins that are often insufficient in notifying users. Some of these plugins require payment for timely email notifications. Lastly, plugin developers make errors during plugin development, such as guideline violations, security vulnerabilities, outdated compatibility, and malicious code inclusion. To address RQ3, it is important to consider that, despite not many security plugins being efficient, as noted by respondents, WordFence is the most commonly used and one of the most effective. Additionally, when selecting plugins, web developers should be cognizant of the fact that relying solely on ratings and reviews is unreliable. Lastly, it is crucial for plugin developers to prioritise security practices to prevent plugin suspensions and patch delays. Table 8.1 in the following section outlines the project objectives and how they were achieved.

Table 8.1 Measuring the aims and objectives.

No.	Objective	How did the research meet the objective?
1	Identify the causes of patch delays and plugin suspensions in WordPress Plugin Development and critically evaluate their impact in previous cases.	As answered by web developers, delays in releasing updates can occur due to technical complexities, compatibility issues, or unforeseen events, requiring plugin developers to address both internal and external factors. Reasons for plugin suspension include security vulnerabilities, guideline violations, poor user experience, outdated compatibility, and containing malicious code.
2	Carry out qualitative research on experienced individuals or business representatives to compose an analytical report.	Three qualitative surveys were conducted among experienced individuals, including freelance and agency web and plugin developers selected from the United Kingdom, United States, and the European Union. Additionally, surveys were conducted among small and SMEs. The overall insights gained from these surveys enabled the creation of the report.
3	Design and implement a proof-of-concept software that solves parts or the entire problem, meeting the demographic group criteria.	A software concept has been developed specifically for SMEs to notify their outsourced web developers, which can significantly mitigate the majority of risks associated with vulnerability due to patch delays. Although it does not solve the issue of patch suspension entirely, it does address a part of the problem by

comparing plugins with WordPress API to detect available updates and provide alerts.

Modifications were made to the initial objectives outlined in the project proposal. Originally, the project aimed to conduct interviews with web and plugin developers to gain insight into the causes of plugin patch delays. While interviews were initially considered a more effective approach, time constraints posed challenges for developers who had existing work and commitments. Consequently, the project was modified to use qualitative surveys instead, which were deemed more practical to collect and had the added benefit of reducing sampling and population bias.

Overall, the project has achieved all the research objectives and goals, providing a comprehensive response to the research question. While many researchers have examined the effectiveness of security plugins, the efficacy of advanced functionalities such as notification systems has been overlooked. This study addresses this gap suggesting several mitigation methods. Firstly, awareness of effective tools and practices among web developers can decrease suspensions. In addition, secure programming techniques can be promoted to plugin developers. If such suspensions cannot be avoided, the utilization of the concept tool may effectively reduce the delay in applying patches in a timely manner.

8.1 Future Work

The dissertation has achieved its aims and objectives, providing valuable insights into the causes of WordPress plugin patch delays and suspensions. The research has identified that outsourcing web development and selecting plugins based solely on ratings and reviews are common mistakes made by SMEs and web developers, respectively, which leave them vulnerable to patch delays and suspensions. Additionally, the research has identified several factors that contribute to patch delays and suspensions, including technical complexities, compatibility issues, and guideline violations.

The implementation of the proof-of-concept software has demonstrated that setting up a notification system for SMEs to alert their outsourced web developers can substantially mitigate most risks associated with becoming vulnerable due to patch delays. The software detects at least 83.3% of available patches and can be a useful tool to reduce the delay in applying patches in a timely manner. However, it should be noted that premium plugins such as "Elementor Pro" cannot be brought to the attention of users due to the absence of such features in the WordPress API.

However future work could focus on several areas that may improve the project. Firstly, it may be beneficial to conduct further research from a socio-economic perspective to explore the behaviour of developers in a broader context. This could involve reviewing literature that examines the real impact of respondents and case studies that focus on software in general. Secondly, implementation of the WPP Monitor plugin could be improved by incorporating an actual vulnerability database and resolving any current bugs such as the monthly option not working. Additionally, the plugin could be updated to check for suspended plugins that are not listed on the directory anymore, as relying solely on authors to alert users of this problem may not be sufficient. Furthermore, future work could explore the possibility of incorporating the WPP Monitor plugin into the WordPress core, although this may come with limitations such as adding more overhead and requiring more storage space which could concern users with limited storage. Overall, further research and improvements to the WPP Monitor plugin could help reduce the risks associated with WordPress plugin patch delays and suspensions.

Bibliography

- Aborujilah, A., Adamu, J., Shariff, S. M., & Long, Z. A. (2022). Descriptive Analysis of Built-in Security Features in Web Development Frameworks. *Proceedings of the 2022 16th International Conference on Ubiquitous Information Management and Communication, IMCOM 2022*. <https://doi.org/10.1109/IMCOM53663.2022.9721750>
- Berghel, H. (2017). Equifax and the Latest Round of Identity Theft Roulette. *Computer*, 50(12). <https://doi.org/10.1109/MC.2017.4451227>
- Cabot, J. (2018). Word press: A content management system to democratize publishing. In *IEEE Software* (Vol. 35, Issue 3). <https://doi.org/10.1109/MS.2018.2141016>
- Fragulis, G. F., Lazaridis, L., Papatsimouli, M., & Skordas, I. A. (2018). O.D.E.S.: An Online Dynamic Examination System based on a CMS Wordpress plugin. *South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference, SEEDA_CECNSM 2018*. <https://doi.org/10.23919/SEEDA-CECNSM.2018.8544928>
- Gall, R. (2022). PSA: Zero-Day Vulnerability in WPGateway Actively Exploited in the Wild. In *WordFence*. <https://www.wordfence.com/blog/2022/09/psa-zero-day-vulnerability-in-wpgateway-actively-exploited-in-the-wild/#:~:text=On%20September%208%2C%202022%2C%20the,sites%20running%20the%20WPGateway%20plugin.>
- Gall, R. (2023). *The Wordfence 2022 State of WordPress Security Report*. <https://www.wordfence.com/wp-content/uploads/2023/01/The-Wordfence-2022-State-of-WordPress-Security-Report.pdf>
- Kasturi, R. P., Fuller, J., Sun, Y., Chabklo, O., Rodriguez, A., Park, J., & Saltaformaggio, B. (2022). Mistrust Plugins You Must: A Large-Scale Study Of Malicious Plugins In WordPress Marketplaces. *31st USENIX Security Symposium*. https://www.usenix.org/system/files/sec22summer_kasturi.pdf
- Knauss, D. (2022). *The 2022 WordPress Vulnerability Annual Report*. <https://ithemes.com/wp-content/uploads/2023/01/WordPress-Vulnerabilities-2022-Annual-Report-Infographic.pdf>
- Mesa, O., Vieira, R., Viana, M., Durelli, V. H. S., Cirilo, E., Kalinowski, M., & Lucena, C. (2018). Understanding vulnerabilities in plugin-based web systems: An exploratory study of WordPress. *ACM International Conference Proceeding Series*, 1. <https://doi.org/10.1145/3233027.3233042>
- Mullenweg, M. (2004, May 22). *WordPress 1.2*. <https://wordpress.org/news/2004/05/heres-the-beef/>
- Murphy, D. T., Zibran, M. F., & Eishita, F. Z. (2021). Plugins to detect vulnerable plugins: An empirical assessment of the security scanner plugins for WordPress. *2021 IEEE/ACIS 19th International Conference on Software Engineering Research, Management and Applications, SERA 2021*. <https://doi.org/10.1109/SERA51205.2021.9509274>
- Patchstack. (2022). *State Of WordPress Security In 2021*. <https://patchstack.com/wp-content/uploads/2022/03/Patchstack-%E2%80%93-State-Of-WordPress-Security-In-2021.pdf>
- Sen, R., Choobineh, J., & Kumar, S. (2020). Determinants of Software Vulnerability Disclosure Timing. *Production and Operations Management*, 29(11). <https://doi.org/10.1111/poms.13120>
- Stöver, A., Gerber, N., Pridöhl, H., Maass, M., Bretthauer, S., Spiecker gen. Döhmman, I., Hollick, M., & Herrmann, D. (2023). How Website Owners Face Privacy Issues: Thematic Analysis of Responses from a Covert Notification Study Reveals Diverse Circumstances and Challenges. *Proceedings on Privacy Enhancing Technologies*, 2023(2). <https://doi.org/10.56553/popets-2023-0051>

- Sucuri. (2021). *2021 Website Threat Research Report*. <https://sucuri.net/wp-content/uploads/2022/04/sucuri-2021-hacked-report.pdf>
- W3Techs. (2023). *Usage Statistics of Content Management Systems*. https://w3techs.com/technologies/overview/content_management
- Williams, B., Damstra, D., & Stren, H. (2020). *Professional WordPress Plugin Development* (Third Edition). Wrox. <https://jicc.org.il/wp-content/uploads/2016/07/professional-wordpress-design-and-development-3rd-edition.pdf>
- WordPress. (2023, March 6). *WordPress Requirements*. <https://en-gb.wordpress.org/about/requirements/>

Appendices

Appendix A Data Extractor

```
import os
import json

cwe_counts = {}
total_cve_count = 0

for filename in os.listdir():
    if filename.startswith('nvdcve-1.1-') and filename.endswith('.json'):
        with open(filename) as f:
            data = json.load(f)

            file_cve_count = 0
            for item in data['CVE_Items']:
                description = item['cve']['description']['description_data'][0]['value'].lower()
                if 'wordpress' in description and 'plugin' in description:
                    problem_types = item['cve']['problemtype']['problemtype_data'][0]['description']
                    for problem_type in problem_types:
                        cwe = problem_type['value']
                        if cwe in cwe_counts:
                            cwe_counts[cwe] += 1
                        else:
                            cwe_counts[cwe] = 1
                    file_cve_count += 1
                    total_cve_count += 1
                    cve_id = item['cve']['CVE_data_meta']['ID']
                    #print(f"{filename}: CVE ID {cve_id} has 'WordPress' and 'Plugin' in description")
                print(f"{filename}: CVEs with 'WordPress' and 'Plugin' in description: {file_cve_count}")

print(f"\nTotal CVEs with 'WordPress' and 'Plugin' in description: {total_cve_count}")
total_cwe_count = sum(cwe_counts.values())
for cwe, count in sorted(cwe_counts.items(), key=lambda x: x[1], reverse=True):
    percentage = round((count / total_cwe_count) * 100, 2)
    print(f"{cwe}: {count} ({percentage}%)")
```

Listing A.1 Data Extraction: h.py

Appendix B Concept Software

```
<?php
/*
 * Plugin Name: WPP Monitor
 * Plugin URI:  https://wppmonitor.com
 * Description: A WordPress Patch Monitor Plugin. Get alerts straight to
 your phone. Avoid Becoming Vulnerable. - An Ethical Hacking & Cybersecurity
 at Coventry University Dissertation.
 * Requires at least: 5.2
 * Requires PHP: 7.2
 * Version:      1.2.0
 * Author:       WPP Monitor
 * Author URI:   https://wppmonitor.com
 * License: GPLv2 or later
 */

// Include Scanning Function
include(plugin_dir_path(__FILE__) . 'inc/engine/scan.php');

// Include Cron Job Function
include(plugin_dir_path(__FILE__) . 'inc/engine/job.php');

// Include Email Function
include(plugin_dir_path(__FILE__) . 'inc/engine/email.php');

// Hook to trigger the scheduled event
add_action('wpp_scan_event', 'job', 0);

function WPP_Monitor()
{
    /* ARGS (Title of the Page, Admin Menu Title, Permission it Has, Slug
 / URL, Call Back Function, Icon for the Menu, Position in the Menu) */
    add_menu_page('WPP Monitor', 'WPP Monitor', 'manage_options',
 'wpp_monitor_slug', 'wpp_monitor_callback', 'dashicons-shield', 999);
}
add_action('admin_menu', 'WPP_Monitor');

/* FUNC Paths for Callbacks */
function wpp_monitor_callback()
{
    include (plugin_dir_path( __FILE__ ) . 'templates/wpp_monitor.php');
}
```

Listing B.1 Core: WPP.php

```
<?php

function send_email($name, $email, $dev_name, $dev_email,
$outdated_plugins){

    // Get the website title
    $site_title = wp_title('|', false, 'right') . get_bloginfo('name');

    // Set the email subject
    $subject = $site_title . " - Critical Level Vulnerability Found at " .
date('h:i A');

    // Set the email message
    $message = "Dear " . $dev_name . "<br><br> Would you be kind and look
at my " . $site_title . " website because at " . date('h:i A') . ", it has
reported as high vulnerability. I have the following outdated plugins:";

    if (!empty($outdated_plugins)) {
        // Add a list of outdated plugins if found
        $message .= "<br><ul>";
        foreach ($outdated_plugins as $plugin) {
            $message .= "<li>" . "<b style='color: red;'>" .
$plugin['name'] . "</b>" . ' (' . $plugin['version'] . ') <a href="' .
$plugin['disable_link'] . '">Disable Plugin</a></li>';
        }
        $message .= "</ul>";
    } else {
        // Add a message if no outdated plugins found
        $message .= "<br><strong>No outdated plugins found!</strong>";
    }

    $message .= "<br> Thank you for checking it! <br><br> " . "<b>" . $name
. "</b>";

    // Set email headers
    $headers = "From:" . $email . "\r\n" .
        "Reply-To:" . $email . "\r\n" .
        "X-Mailer: PHP/" . phpversion() . "\r\n" .
        "Content-type:text/html;charset=UTF-8" . "\r\n";

    // Send the email and display a message
    if(mail($dev_email, $subject, $message, $headers)){
        // Show a success message to the user
        echo '<div class="notice notice-success is-dismissible"><p>Email
sent to ' . $dev_email . '</p></div>';
    } else {
        // Show a success message to the user
        echo '<div class="notice notice-error is-dismissible"><p>Email
failed to send to ' . $dev_email . '</p></div>';
    }
}
```

Listing B.2 Engine: email.php

```
<?php

function job(){
    // Getting the stored values.
    $name = get_option('my_plugin_name_name');
    $email = get_option('my_plugin_name_email');
    $dev_name = get_option('my_plugin_name_dev_name');
    $dev_email = get_option('my_plugin_name_dev_email');

    //Send the attributes and scan site.
    $outdated_plugins = scan_function();
    send_email($name, $email, $dev_name, $dev_email, $outdated_plugins);
}
```

Listing B.3 Engine: job.php

```
<?php

function scan_function(){

    // Check for outdated plugins
    $plugins = get_plugins(); // Get all installed plugins
    $outdated_plugins = array(); // Initialize an array for storing
    outdated plugins

    // Loop through each plugin and check if it's outdated
    foreach ($plugins as $plugin_path => $plugin) {
        $plugin_data = get_plugin_data(WP_PLUGIN_DIR . '/' .
$plugin_path); // Get plugin data
        $plugin_version = $plugin_data['Version']; // Get plugin version

        // Get WordPress plugin list in JSON and compare with current
        version
        $response =
wp_remote_get('https://api.wordpress.org/plugins/info/1.0/' .
$plugin_data['Name'] . '.json');

        if (!is_wp_error($response)) {
            $body = wp_remote_retrieve_body($response); // Retrieve
            response body
            $plugin_info = json_decode($body); // Parse JSON data

            // Check if the plugin is outdated and add to array if it is
            if (version_compare($plugin_version, $plugin_info->version,
'<')) {
                $outdated_plugins[] = array(
                    'name' => $plugin_data['Name'],
                    'version' => $plugin_version,
                    'disable_link' =>
wp_nonce_url(admin_url('plugins.php?action=deactivate&plugin=' .
$plugin_path), 'deactivate-plugin_' . $plugin_path)
                );
            }
        }
    }
    return $outdated_plugins; // Return the array of outdated plugins
}
```

Listing B.4 Engine: scan.php

```
<?php
    // Set a default tab
    $default_tab = 'active-scans';
    // Check if a tab is specified in the URL parameters, otherwise use
    the default tab
    $tab = isset($_GET['tab']) ? $_GET['tab'] : $default_tab;
?>
<div class="wrap">
    <h1>Dashboard</h1>
    <p><b>Introducing WPP Monitor!</b> The Scanner WordPress Plugin that
    scans for patches!</p>

    <!-- Navigation tabs for different sections of the admin page -->
    <nav class="nav-tab-wrapper">
        <!-- The active-scans tab is highlighted if the $tab variable is
        set to 'active-scans' -->
        <a href="?page=wpp_monitor_slug" class="nav-tab <?php
        if($tab=='active-scans'):?>nav-tab-active<?php endif; ?>">Active
        Scans</a>
        <!-- The integration tab is highlighted if the $tab variable is
        set to 'integration' -->
        <a href="?page=wpp_monitor_slug&tab=integration" class="nav-tab
        <?php if($tab=='integration'):?>nav-tab-active<?php endif;
        ?>">Integration</a>
        <!-- The settings tab is highlighted if the $tab variable is set
        to 'settings' -->
        <a href="?page=wpp_monitor_slug&tab=settings" class="nav-tab <?php
        if($tab=='settings'):?>nav-tab-active<?php endif; ?>">Settings</a>
    </nav>

    <!-- The content of the currently active tab is displayed here -->
    <div class="tab-content">
        <?php
            // Include the appropriate PHP file depending on the current
            tab.
            switch($tab):
                case 'active-scans':
                    include (plugin_dir_path( __FILE__ )
                    'manager/active_scans.php');
                    break;
                case 'settings':
                    include (plugin_dir_path( __FILE__ )
                    'manager/settings.php');
                    break;
                case 'integration':
                    include (plugin_dir_path( __FILE__ )
                    'manager/integration.php');
                    break;
            endswitch;
        ?>
    </div>
</div>
```

Listing B.5 Manager: wpp_monitor.php

<p>In this section you can delete and see which scanning is on.</p>

```
<?php
$crons = _get_cron_array();

if (isset($_POST['wpp_delete_event_nonce'])) {
    $nonce = $_POST['wpp_delete_event_nonce'];
    if (wp_verify_nonce($nonce, 'wpp_delete_event')) {
        $timestamp = $_POST['wpp_delete_event_timestamp'];
        $hook = $_POST['wpp_delete_event_hook'];
        unset($crons[$timestamp][$hook]);
        _set_cron_array($crons);
    }
}

// Filter the cron jobs to only show those with the name 'wpp_scan_event'
$filtered_crons = array_filter($crons, function($cron) {
    return isset($cron['wpp_scan_event']);
});

if (!empty($filtered_crons)) {
    echo '<table class="wp-list-table widefat fixed striped">';
    echo '    <thead><tr><th>Hook                Name</th><th>Frequency                in
Seconds</th><th>Next Run</th><th>Arguments</th><th></th></tr></thead>';
    echo '    <tbody>';
    foreach ($filtered_crons as $timestamp => $cron) {
        foreach ($cron as $hook => $details) {
            //Filter any cron jobs under the specified name.
            if ($hook === 'wpp_scan_event') {
                echo '    <tr>';
                echo '    <td>' . $hook . '</td>';
                foreach ($details as $key => $value) {
                    echo '    <td>' . ucfirst($value['interval']) . ' (' .
ucfirst($value['schedule']) . ') ' . '</td>';
                }
                echo '    <td>' . date('Y-m-d H:i:s', $timestamp) . '</td>';
                echo '    <td>';
                foreach ($details as $key => $value) {
                    foreach ($value['args'] as $argKey => $argValue) {
                        echo $argKey . ': ' . $argValue . ', ';
                    }
                }
                echo '    </td>';
                echo '    <td><form method="post">';
                echo '    <input type="hidden" name="wpp_delete_event_nonce"
value="" . wp_create_nonce('wpp_delete_event') . '">';
                echo '    <input                                type="hidden"
name="wpp_delete_event_timestamp" value="" . $timestamp . '">';
                echo '    <input type="hidden" name="wpp_delete_event_hook"
value="" . $hook . '">';
                echo '    <input type="submit" value="Delete">';
                echo '    </form></td>';
                echo '    </tr>';
            }
        }
    }
    echo '</tbody>';
}
```

```
        echo '</table>';
    } else {
        echo '<p>No cron jobs found with the name "wpp_scan_event"</p>';
    }
?>
<br><button                class="button                button-primary"
onclick="window.location.reload();">Refresh</button>
```

Listing B.6 Manager: active_scans.php

```
<?php

// Function to display the integrations section in the settings page
function wpp_monitor_settings_integrations() {
    ?>
    <div class="wrap">
        <p>In this section you can add the essential services to your
        security. Please add your API Keys.</p>
        <form method="post" action="options.php">
            <?php settings_fields('data_storage_options'); ?>
            <?php do_settings_sections('data-storage'); ?>

            <!-- WPScan Application API -->
            <h2>WPScan Database API</h2>
            <table class="form-table">
                <tr valign="top">
                    <th scope="row">API Key</th>
                    <td>
                        <input type="text" name="api-key" value="<?php
                        echo esc_attr(get_option('api-key')); ?>" /><br>
                        <p>To integrate with our forms you need an API Key.
                        <a target="_blank" href="https://wpscan.com/"><b>Generate</b></a></p><br>
                        <div bis_skin_checked="1"><button data-
                        action="api_key_validate" data-nonce="4bfbd14751" class="button"
                        id="api_key_button">Validate API Key</button></div>
                    </td>
                </tr>
            </table>
            <hr>
            <?php submit_button(); ?>
        </form>
    </div>
    <?php

}

// Function to register the settings
function register_settings() {
    register_setting('data_storage_options', 'api-key');
}

// Function to store the data
function store_data() {
    $value = get_option('api-key');
    $this->data[$key] = $value;
}

// Function to retrieve the data
function retrieve_data($key) {
    if (array_key_exists($key, $this->data)) {
        return $this->data[$key];
    }
    return null;
}

// Call the function to display the integrations section
$data_storage = wpp_monitor_settings_integrations();
```

```
// Hook the function to register the settings
add_action('admin_init', array($data_storage, 'register_settings'));

// Hook the function to store the data
add_action('update_option_data_api_key', array($data_storage,
'store_data'));
```

Listing B.7 Manager: integration.php

```
<?php
// Register activation hook
register_activation_hook(__FILE__, 'save_function');
// Save the form data to the database
function save_function() {
    // Retrieve the form data
    $name = isset($_POST['name']) ? sanitize_text_field($_POST['name']) :
    '';
    $email = isset($_POST['email']) ? sanitize_email($_POST['email']) : '';
    $dev_name = isset($_POST['web-name']) ? sanitize_text_field($_POST['web-
name']) : '';
    $dev_email = isset($_POST['web-email']) ? sanitize_email($_POST['web-
email']) : '';
    $frequency = isset($_POST['frequency']) ?
    sanitize_text_field($_POST['frequency']) : '';
    // Save the form data to the WordPress database
    update_option('my_plugin_name_name', $name);
    update_option('my_plugin_name_email', $email);
    update_option('my_plugin_name_dev_name', $dev_name);
    update_option('my_plugin_name_dev_email', $dev_email);
    update_option('my_plugin_name_frequency', $frequency);
    // Schedule the cron job based on the selected frequency
    switch ($frequency) {
        case 'minute':
            wp_schedule_event(time(), 'minute', 'wpp_scan_event', array($name,
$email, $dev_name, $dev_email));
            break;
        case 'daily':
            wp_schedule_event(time(), 'daily', 'wpp_scan_event', array($name,
$email, $dev_name, $dev_email));
            break;
        case 'weekly':
            wp_schedule_event(time(), 'weekly', 'wpp_scan_event', array($name,
$email, $dev_name, $dev_email));
            break;
        case 'monthly':
            wp_schedule_event(time(), 'monthly', 'wpp_scan_event', array($name,
$email, $dev_name, $dev_email));
            break;
    }
    // Show a success message to the user
    echo '<div class="notice notice-success is-dismissible"><p>Your settings
have been saved.</p></div>';
}

// Save or Scan?
if (isset($_POST['save'])) {
    //Save the whatever is in input.
    save_function();
} elseif (isset($_POST['scan'])) {
    // Trigger the scan immediately (leaving out cron).
    job();
}
?>

<!-- Display the form to the user -->
<form method="post">
```

```

<!-- WEBSITE'S OWNER'S -->
<h4>Your Details</h4>
<label for="name">Your Name</label>
<input type="text" name="name" id="name" required value="<?php echo
esc_attr(get_option('my_plugin_name_name')); ?>"><br><br>
<label for="email">Your Email</label>
<input type="email" name="email" id="email" required value="<?php echo
esc_attr(get_option('my_plugin_name_email')); ?>"><br><br>
<!-- DEVELOPER'S SECTION -->
<h4>Developer's Details</h4>
<label for="web-name">Developer's Name</label>
<input type="text" name="web-name" id="web-name" required value="<?php
echo esc_attr(get_option('my_plugin_name_dev_name')); ?>"><br><br>
<label for="web-email">Developer's Email</label>
<input type="email" name="web-email" id="web-email" required
value="<?php echo esc_attr(get_option('my_plugin_name_dev_email'));
?>"><br><br>
<!-- SCAN FREQUENCY -->
<h4>Scan Frequency</h4>
<select name="frequency" id="frequency">
  <option value="minute" <?php
selected(get_option('my_plugin_name_frequency'), 'minute'); ?>>1
Minute</option>
  <option value="daily" <?php
selected(get_option('my_plugin_name_frequency'), 'daily');
?>>Daily</option>
  <option value="weekly" <?php
selected(get_option('my_plugin_name_frequency'), 'weekly');
?>>Weekly</option>
  <option value="monthly" <?php
selected(get_option('my_plugin_name_frequency'), 'monthly');
?>>Monthly</option>
</select><br><br>
<!-- SUBMIT BUTTON -->
<input class="button button-primary" type="submit" name="save"
value="Save Settings">
<input class="button button-secondary" type="submit" name="scan"
value="One Click Check">
</form>

```

Listing B.8 Manager: settings.php