Received: 10 February 2023 | Revised: 2 May 2023 | Accepted: 24 May 2023 | Published online: 31 May 2023

REVIEW

Artificial Intelligence and Applications
2023, Vol. 1(4) 214–220
DOI: 10.47852/bonviewAIA3202754

A Study of Vulnerability Scanners for Detecting SQL Injection and XSS Attack in Websites

Seema Sharma[1,*]

[1]*Department of Computer Science and Engineering, JECRC University, India*

Abstract: In the modern world, the internet makes our lives easier. Making use of online services like social media, online banking, and online shopping can provide information while also saving time and resources. As the user's information is then accessible on the website, maintaining its security is essential. To determine whether a site is susceptible or not, a website vulnerability scanner is utilized. A web attack can happen very quickly if the website is weak. Due to this, data theft is possible. This paper aims to identify the website's weaknesses and vulnerabilities and make improvements. If a website is vulnerable, attacks can be carried out quickly. It allows for data theft. This paper's objective is to discover the website's vulnerabilities and sources of vulnerability so that they can be fixed. If the web app is weak, an attacker could exploit the website by sending in malicious code from the client side. To resolve these problems, this paper outlines potential modern best practices. Also, it is beneficial for web security researchers.

Keywords: XSS, SQL, injection attack, vulnerability, security

1. Introduction

Website vulnerability scanners are used to identify flaws in websites and then display the types of bugs found. Vulnerability scanners are important tools for identifying as well as reporting security flaws in the IT infrastructure of an organization. This is an essential security procedure that every organization may use. These scans can provide details about possible security vulnerabilities, which can help an organization has a better understanding of the security hazards that might be present in their background (Jain et al., 2022).

To ensure that they are getting complete coverage of each asset and to build a complete view, many organizations employ multiple vulnerability scanners (Fu et al., 2016). Over time, many different scanners have been developed with a variety of features and possibilities. Cybersecurity threats increase for two reasons: strategically and practically. Six main areas can be used to classify vulnerabilities: network, personnel, software, organizational, physical, and hardware. The vulnerability can occur due to several factors, such as complex software, weak device connectivity, weak password handling, software defects, and uncontrolled client-side input. To make our website consistently secure, we have conducted a lot of analysis on various kinds of websites linked to PHP, HTML, JavaScript, and CSS (Wang et al., 2018).

One of the attacks that occur on vulnerable web applications is an injection attack. An injection attack is a technique where an attacker inserts or infects harmful code into your app to steal your details or harm your system (PHP Labware, 2023). Your software went to the request without question, believing that you issued the request. The injection attack is among the oldest and most destructive cyberattacks. Once the malicious codes are injected, an attacker can hijack your network and obtain any information they require from it (Sharma & Yadav, 2021; Zukran & Siraj, 2021).

The most common injection attacks are SQL injection (SQLI) and cross-site scripting (XSS) injection attacks (Liu et al., 2023; Buja et al., 2022). When initiating a SQLI, a SQL command is used to send database queries, particularly ones that save, access, retrieve, or delete database data. By using your comment areas, form input fields, or other user-accessible channels, the attacker manipulates your SQL by targeting it. In other words, SQL queries can pass directly to and query the database because of the fields that are open for user input. Web-based attacks using SQLI are mostly uncontested by firewalls and other systems for intrusion detection.

In short, SQLI is generated because the fields available for user input allow SQL statements to pass directly to and query the database. The security against extensive SQLI web attacks offered by firewalls and similar intrusion detection systems is minimal to zero. Another one is the XSS injection attack. It is one of the most frequent application-layer web attacks, focusing on scripts that are injected into pages but run on the server rather than on the client.

Client-side scripting languages like HTML and JavaScript have cybersecurity flaws that make XSS a hazard. The idea behind XSS is to trick a client-side script in a web application to run the way a malicious user wants it to. The rest of this paper is divided as follows: Section 2 provides the motivation and objective of the study, Section 3 provides a review, Section 4 shows the implementation, Section 5 discusses the experimental results, and Section 6 provides a conclusion

*Corresponding author: Seema Sharma, Department of Computer Science and Engineering, JECRC University, India. Email: seemasharmacg@gmail.com

2. Motivation

Attacks utilizing input injection can serve several purposes inside the present system. They are typically used by malicious users as a method of gaining unauthorized access to confidential data from a back-end database or of uploading malicious code to a fake server that would then spread malware to innocent users. This may lead to these customers learning about their credentials or personal information, which would be exciting for them (Vandana et al., 2014; Biswas & Majumder, 2014). This approach mainly replaces the existing time-consuming, expensive, and low-interactive structure for cognitive scanning procedures. The major features of the approaches are reputation maintenance and the detection of several unusual vulnerabilities, data storage for scanning, program design, and the generation of reports for all scanned websites. Some advantages of the suggested framework include the quickest Web crawler, the simplest sessions to access, scanning of any specific page, a variety of examinations, the fastest electronic scanner, Google Dorking available, etc. The primary objective of this paper is to provide an overview of the analysis and specifications of the current scenario or system to identify its functioning components. This paper provides a detailed description of the whole software specification of the entire differential vulnerabilities scanner. In this paper, vulnerabilities scanner aims to develop an online scanner for all enterprises and organizations. It is made to be used by developers and will act as the starting point for testing.

3. Review

As new technologies, HTML elements, and JavaScript functions emerge, attacks on web applications are expanding quickly. Aggressors use XSS flaws to inject malicious JavaScript code into the victim's web applications to steal the resources (cookies, credentials, etc.) from the victim's web browsers (Dukes et al., 2013). The vulnerability can be classified as follows: a flaw in input validation (client-side request level), a weakness in session management (session level), and a vulnerability in the application logic (whole application).

Vulnerabilities in input validation occur if an attacker discovers that an application makes unverified assumptions about the nature, length, format, or range of input data, The program is then subject to input validation vulnerabilities. Attackers are ready to introduce maliciously designed inputs into systems if inputs are not correctly sanitized. These inputs may change program functionality or grant unauthorized access to resources. Incorrect input validation can open them to a variety of attacks, including buffer overflow attacks, SQLI attacks, XSS attacks, and other code injection assaults (Dukes et al., 2013). Session management security flaws, a web application, may track user inputs and preserve application states, thanks to session management. The cooperation between the client and the server is used to manage sessions in web application development. To prevent session hijacking, the confidentiality, integrity, and validity of the session ID must be guaranteed because it is the only evidence of the client's identity (Tang et al., 2020). The goal of an injection attack is to exploit a flaw in the program's logic, bypassing or missing the proposed order in which application functions are established. These attacks typically target a website, but they can also be directed toward its users and their personal information.

Traditional SQL infusion barriers employ boycotts to filter out illegal symbols, and a lot is currently done. The book "SQL Injection Attacks and Defence" (Tang et al., 2020) describes two distinct types of information approval strategies to deal with securing SQL infusion. Although entry testing can be used to make up for the shortcomings of the high contrast list separating guard method, it cannot generally fix the flaws (Tian et al., 2012). To test the scanners' ability to distinguish SQLI and XSS, Parvez et al. (2015) dissected the exhibition and identification capabilities of the most modern discovery web application security scanners against hidden SQLI and hidden XSS.

On the one hand, in conventional web applications, the server-side part of the function has increasingly been moved to the client. On the other hand, because current browsers support the HTML API, JavaScript code is becoming more and more common, and its functions are becoming increasingly complicated. All of these result in steadily increasing security issues (Mitropoulos et al., 2017; Sivakorn et al., 2016). The third sort of XSS is also referred to as DOM-XSS, and it is purely a client-side security problem. In other words, it simply causes XSS by parsing the browser's DOM and does not need to be directly participating in the server's analytic response.

Another API (Snow et al., 2016) uses data in the form of executable code. The server is entirely undetectable. Third, a lot of contemporary web apps make use of controlled, server-invisible JavaScript code from third parties.

Hydara et al. (2015) and others examined a significant amount of earlier work on XSS before 2013, but only 0.9% of it was detected in the study about DOM-XSS. Vogt et al. (2007) suggested a static analysis and dynamic information flow tracing method to lessen the danger of XSS. However, instead of tracking sensitive data, they concentrated on sensitive information, such as cookies or prospective information leaks, and were unable to identify the root cause of vulnerabilities.

A Firefox plug-in called Ra.2 (Wang et al., 2018) uses black box fuzzing to detect DOM-XSS. The first DOM-XSS detection tool based on tent tracking, DOMINATOR, was created by changing the Spiderman-key JavaScript engine in Firefox. It cannot, however, implement automatic vulnerability scanning.

A black box fuzzing (Sutton et al., 2007) based on tainted growth was proposed by Saxena et al. (2010). In this study, automated random fuzzing techniques are paired with dynamic tent analysis. The identical prototype tool, FLAX, was created at the same time. To find XSS, Criscione (2013) proposed an automation method based on the black box, and it tested and verified vulnerabilities using a real browser.

The DOMXSS scanner (Gomez, 2016) is used to check the source code of web pages for DOM XSS sources and sinks without finding vulnerabilities. Black box testing, static analysis (Xie & Aiken, 2006), and dynamic analysis are three current directions for DOM-XSS research. The black box test suffers from false negatives as a result of its coverage of the attack vector limitations. The static analysis approach can address simple issues, but when it is used in a complicated situation, it has a high accuracy rate and a high false-positive rate. In this paper, DOM-XCD vulnerability discovery and verification are accomplished using dynamic analysis and dynamic tent tracking technologies.

Zukran & Siraj (2021) suggest the model OW ASP ZAP that examines the vulnerability in the website that has improper inputs. Invalid inputs cause several vulnerabilities. They presented a collection of indicators that tested and handled invalid inputs based on this possibility. To put this paradigm into practice, a tool is created. They evaluated a number of randomly chosen websites to test the model. The drawback of the model is no special authorization or access to any of the tested websites is provided by the tool.

Alsmadi et al. (2021) offer conventional defense systems that employ static and heuristic methods to recognize previously known SQLI attacks. Researchers use machine learning methods that can identify new and previously unidentified attack types.

The author suggests using a probabilistic neural network to recognize SQLI attacks, utilizing deep learning to improve detection accuracy. They used the BAT algorithm, a metaheuristic optimization method, to determine a smoothing parameter's ideal value. To determine a smoothing parameter's ideal value, they used the BAT algorithm, a metaheuristic optimization method. This experiment used a dataset of 6,000 SQLIs and 3500 regular queries. In this experiment, a dataset of 6,000 SQLIs and 3500 regular queries was used.

4. Methods and Tools for Preventing and Detecting Vulnerability

Strategies to classify risks in web use can be classified in more detail as follows:

(I) Static investigation: It is the examination of the source code before executing a program. This methodology makes preparations for the event of particular sorts of vulnerabilities and does not give space for indistinct defenselessness at the hour of coding.
(II) Dynamic investigation: It describes in detail what the program does while running, mainly through the mediator's interface after analyzing the syntactic structure.

Detection methods: The attack detection framework is an innovation that establishes a complex network of double dealings at scale without the need for technical expertise. There are numerous detecting techniques. A few are summarized below.

In the design of the web apps' automatic detection system, analytical taint analysis methods are used to detect persistent attacks. It is preventing further site attacks by crossing the filter pattern as a dynamical pattern filtering method for XSS detection. Its testing is a consistent method for XSS detection. A protection model is created to protect the website from site attacks by the Zen system. It is a nonpersistent approach to detection.

4.1. Tools

OWASP is a society that provides freely available articles, documentation, methodologies, equipment, and technologies within the field of web application safety (Trickel et al., 2022; Dukes et al., 2013).

- N-Stalker: The N-Stalker is a scanner for web application security that examines a web application for numerous vulnerabilities, including those at the application layer and network level (Trickel et al., 2022).
- Acunetix: Acunetix is a security testing automation tool that was founded to combat the rise in attacks at the web application layer.
- Paros: A Java-based HTTP/HTTPS proxy for evaluating web application vulnerability. It supports editing HTTP messages on the fly.
- Web Scarab: This Java-based solution is used to monitor applications that employ the HTTP or HTTPS protocol. This functions as a catch-all intermediary to examine ongoing and active solicitation and approachable site pages. It can recognize various vulnerabilities in web applications like CSRF, XSS, SQLI, and so on (Port Swigger, 2023).
- OWASP Xenotix XSS Exploit Framework: It is the best in the class structure created under OWASP ventures to distinguish and misuse the XSS assault. It includes three fuzzes to limit the checking time and yield better results.
- ImmuniWeb® On-Demand, (n.d.): It is a tool used for multilayer web applications, and it links the capacities of AI and its techniques. It recommends a quick, versatile, and economic strategy for distinguishing vulnerabilities. It addresses all 10 significant flaws listed by OWASP (HTML purifier, n.d.).
- HTML Purifier: It is a quick and straightforward HTML sanitizer created in Java under OWASP ventures. To prevent an XSS attack, it only permits HTML written by external apps (Snyder, 2019).
- htmLawed: For the program to access and create the HTML labels and features permitted by the site management, it is written in PHP to separate the HTML content. It is speedy, flexible, and uses minimum memory (OWASP Projects, 2014).
- XSSer: This is a scripter, a programmed framework to recognize, abuse, and inform about XSS vulnerabilities present in web applications.
- Burp Scanner: It is an entirely robotized access analyzer that security specialists utilize to test an application. It tends to be incorporated with different procedures to get compelling outcomes (on-demand).

4.2. Defenses approaches

There are a variety of safety efforts available to secure web applications against XSS attacks (Rocha and Souto, 2014). Avoid executing scripts on websites with vulnerable content. The web application may be in the safe harbor can pick one of the accompanying approaches.

- **Content filtering:** Before transmitting poor HTML to a browser, an application may make an effort to extract and remove any text and script from it.
- **Browser collaboration:** The system can communicate with the web browser by allowing the browser to know which documents have been shown on the web page while still preserving the authorization rules.
- **Well-known prevent XSS:** There are many ways to ensure the targeting of site writing currently, such as user input validation, sanitizing user input, and the exercise of a content protection policy.
- **Sanitize user input:** Sanitizing user input, for example, G.E.T. solicitations and treats, will quickly place you in a prime spot against XSS assaults. This resistance strategy works on sites that allow HTML markup to require an information script to clear inappropriate or harmful user entries.
- **Validate user input:** Information approval is the way to test all client or application inputs and block inaccurately framed information from entering a data framework. This OWASP deception sheet maintains that client input approval is not a silver-bullet answer for XSS prevention; however, it can help keep clients from importing unusual characters into top-down fields in the form.
- **Utilize a content security policy:** By allowing only specific types of content from trusted sources, the content security strategy helps define guidelines to prevent illegal content. The content safety mode teaches the user browser only to enable the content to be used.

5. Implementation

The hardware and software used in its compilation are the main factors that determine whether a work will run successfully. The hardware that is employed in the simple machine must be capable of supporting the software that will be mounted for assembling the paper. This essay discusses the hardware and software that are easily accessible on each machine that a user is given.

Hardware requirement: CPU: i3 or above; ram: 500 MB or above; HDD: 20 GB or above

Software requirement: Bone: 8 senses of the window, macOS, any Linux-based distro.

Broadcast: Python 2.7 or above.

Language: Python (Python is a physical object-oriented, high-level programming language with integrated dynamic semantics primarily for web and app development).

Library required:

(i) Urllib2: The urllib2 mental faculty defines maps and classes that aid in identifying URLs outside a complicated environment using simple and digest authentication, redirections, cookies, and more.

(ii) SYS: This faculty gives users access to some variables that the interpretive program uses or maintains, as well as mapping that has a close relationship with the interpreter.

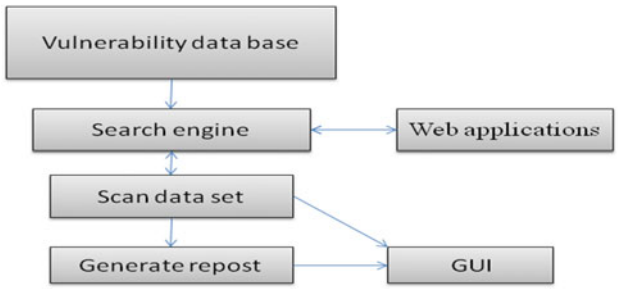
(iii) OS: This module offers a portable approach to using functionality that depends on the operating system being used. Open() can be used to simply read or write a file. The os. path module can be used to alter paths. The file input module can be used to read all of the short letters in all of the program line files.

6. Framework and Working

Vulnerability scanners are programs that analyze the architecture of a network, disclose found vulnerabilities, and provide recommendations on how to fix them (Wang et al., 2018). Figure 1 illustrates the fundamental structure of a vulnerability scanner. Frequent vulnerability assessments are essential for maintaining robust cybersecurity. An organization is nearly certain to have at least one unpatched vulnerability that puts it at risk due to the complete number of vulnerabilities that exist and the complexity of the typical company's digital infrastructure. Discovering these weaknesses before an attacker can strike can mean the difference between a successful attack and an unpleasant one. Vulnerability scans can be carried out from either inside or outside the network or the network segment that is being assessed. The vulnerability of servers and applications that are directly accessible via the internet can be assessed by organizations by conducting external scans from outside the boundaries of their network.

The steps for vulnerability assessment are as

Figure 1
Process of a vulnerability scanner



follows:

Resource identification: Making this decision is not always as easy as it sounds because you have to determine what you want to scan. One of the most common cybersecurity concerns that enterprises face is a lack of visibility into their digital infrastructure and linked devices.

Vulnerability scanning: It is also known security flaws can be found using vulnerability scanners, which also offer advice on how to address them. It is possible to find a lot of information on susceptible software because these vulnerabilities are frequently reported to the public. The infrastructure of an organization uses these data to find weak hardware and software using vulnerability scanners. First, the scanner sends probes to systems to identify: In addition, the scanner sends certain probes to discover particular vulnerabilities, which can only be tested by delivering a safe exploit that verifies the weakness is present. These kinds of probes can spot widespread flaws like "command injection," "XSS," or the use of systems' default users and passwords.

The report analysis:- The scanner offers a report on the assessment following the conclusion of the vulnerability scan. The following factors consist in the report and create remedial plans based on it:

- Severity: The severity of a suspected vulnerability should be shown by a vulnerability scanner.
- Vulnerability: Internet-facing systems should receive higher priority for remediation since they are more likely to be attacked by any random attacker scouring the internet.

In the modern world, the internet makes our lives easier. Making use of online services like social media, online banking, and online shopping can provide information while also saving time and resources. As the user's information is then accessible on the website, maintaining its security is essential. To determine whether a site is susceptible or not, a website vulnerability scanner is utilized. A web attack can happen very quickly if the website is weak. Due to this, data theft is possible. This paper aims to identify the website's weaknesses and vulnerabilities and make improvements. If a website is vulnerable, attacks can be carried out quickly. It allows for data theft.

This paper's objective is to discover the website's vulnerabilities and sources of vulnerability so that they can be fixed. If the web app is weak, an attacker could exploit the website by sending in malicious code from the client side. To resolve these problems, the resources must be identified, vulnerabilities must be scanned for, and reports must be analyzed, as shown in Figure 1.

7. Test and Results

Testing is the process of running the program to see whether there are any bugs. This is the final step in the verification and verification process. We have also made an effort to fix mistakes made in earlier steps. A solid test case has a strong chance of spotting an error that has not been found yet.

Help page: The help page of the scanner shows the argument such as e for browser information, h for help, t for targeted web side, o for output result, r for revise domain, etc. as shown in Figure 2.

Dump results: The information that was scanned from the server and represented by the dump, such as URL details and the language used to create the website, is included. The result is stored in json file as shown in Figure 3.

Result: The scanner has displayed the outcome of the vulnerable URL including the domain name, server name, and database name. The result of the scanner is shown in Figure 4.

Scanning a website: The output of the scanning result of the vulnerable website is shown in Figure 5, which shows the actual website's service details.

Figure 2
Help page of the scanner

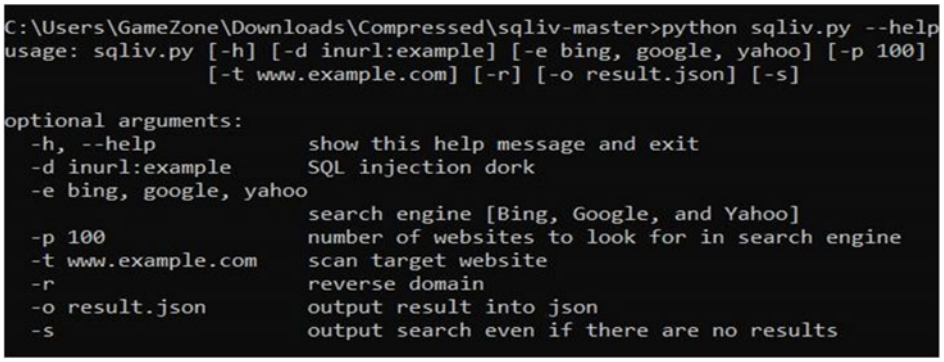


Figure 3
Dump result of the scanner

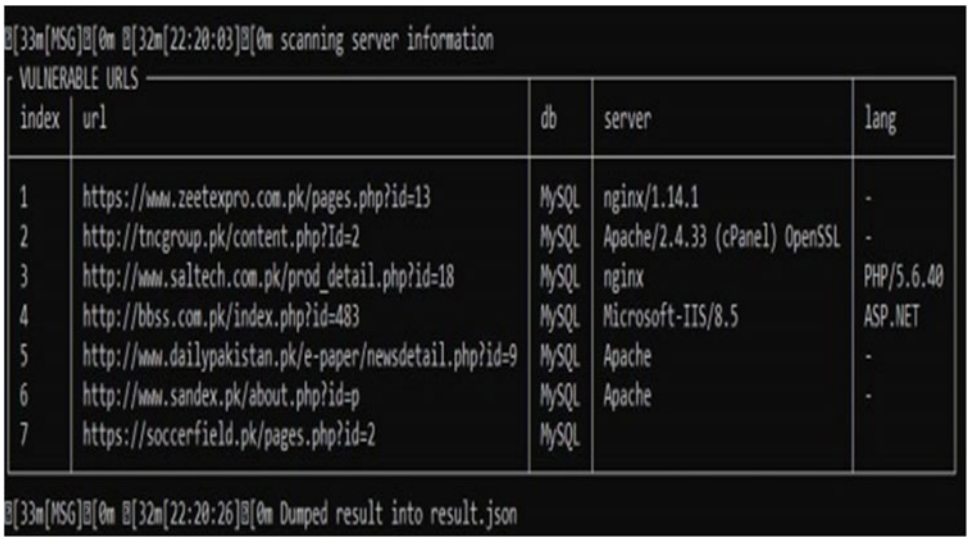


Figure 4
Result of the scanner

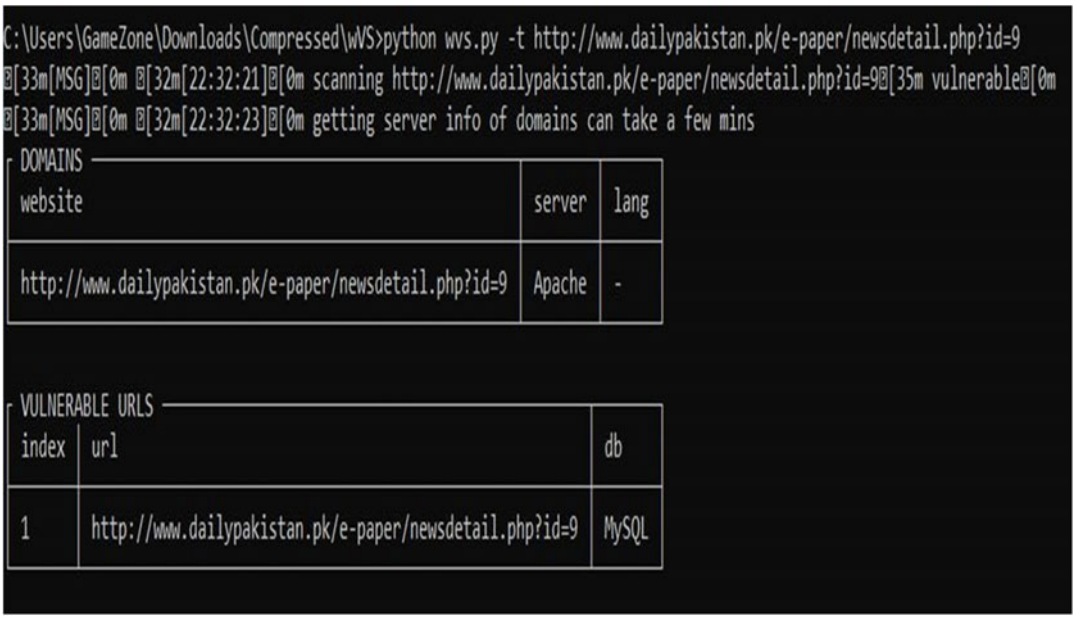
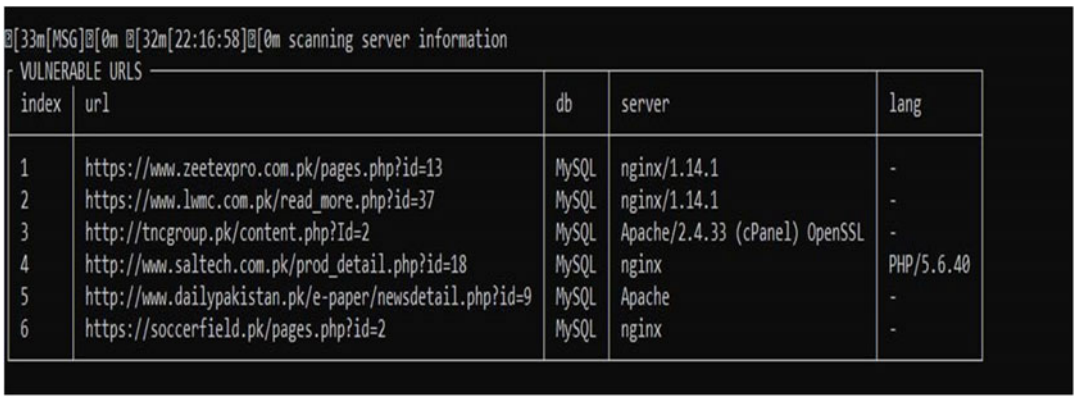


Figure 5
Scanning website



The output of the vulnerability scanner is shown in Figures 1–4. As illustrated in Figure 2, the scanner is used on the Bing, Google, and Yahoo search engines. The server name and vulnerable URL are depicted in Figure 3.

8. Conclusion

There are various security organizations available today to protect the website. In this piece, we employ a vulnerability scanner, scan the website link, and determine whether or not this website is vulnerable. This paper can be very helpful in fixing vulnerabilities on websites. Since the primary focus of the paper is security, the framework's display is appropriate because every objective was met. This paper also offers some advice for the ever-expanding technology industry. The "Site Vulnerability Scanner" was well-planned, created, and tested in this paper. This module has exceptional capabilities; therefore, it can be enhanced to create a complete framework because of this. It offers the protection analyst all the required security issues and how to prevent hackers from exploiting them. This paper aims to provide a security analyst with a thorough overview of all fundamental security issues and a suggestion for how programmers could mitigate them. It provides all the advantages required for users to obtain and personalize the information presented to them. After choosing to use this framework, we believe that the business will eventually realize its significance and worth, as well as the issues with the manual procedure. Vulnerability scanners are anticipated to incorporate new automated solutions as threat environment modifications, helping organizations manage security risks without losing adaptability or speed. These technologies will help organizations stay one step ahead of hackers.

Conflicts of Interest

The author declares that she has no conflicts of interest to this work.

References

Alsmadi, I., AlEroud, A., & Saifan, A. A. (2021). Fault-based testing for discovering SQL injection vulnerabilities in web applications. *International Journal of Information and Computer Security*, *16*(1–2), 51–62. https://doi.org/10.1504/IJICS.2021.117394

Biswas, A., & Majumder, D. (2014). Genetic algorithm based hybrid fuzzy system for assessing morningness. *Advances in Fuzzy Systems*, *2014*, 8. https://doi.org/10.1155/2014/732831

Buja, A., Luma, Z. B., Ademi, R., & Bela, B. (2022). An online SQL vulnerability assessment tool and its impact on SMEs. *International Journal of Advanced Research in Computer Science*, *13*(5).

Criscione, C. (2013). *Drinking the ocean-finding XSS at Google Scale*.

Dwivedi, V., Yadav, H., & Jain, A. (2014). A survey on web application vulnerabilities. *International Journal of Computer Applications*, *108*(1), 25–31.

Dukes, L., Yuan, X., & Akowuah, F. (2013). A case study on web application security testing with tools and manual testing. In *2013 Proceedings of IEEE SoutheastCon*, 1–6.

Fu, Z., Wu, X., Guan, C., Sun, X., & Ren, K. (2016). Toward efficient multi-keyword fuzzy search over encrypted outsourced data with accuracy improvement. In *IEEE Transactions on Information Forensics and Security*, *11*(12), 2706–2716.

Gomez, R. (2016). *DOMXSS scanner*. Retrieved from: https://github.com/yaph/domxssscanner

HTML Purifier. (n.d.). *Standards-Compliant HTML filtering*. Retrieved from: http://htmlpurifier.org/

Hydara, I., Sultan, A. B. M., Zulzalil, H., & Admodisastro, N. (2015). Current state of research on cross-site scripting (XSS)–A systematic literature review. *Information and Software Technology*, *58*, 170–186. https://doi.org/10.1016/j.infsof.2014.07.010

ImmuniWeb® On-Demand (n.d.) *Web application penetration testing*. Retrieved from: https://www.immuniweb.com/products/ondemand/

Jain, A., Aadithyanarayanan, M. R., Anand, A., Maheshwari, H., Gonge, S., Joshi, R., & Kotecha, K. (2022). Web scanner: An innovative prototype for checking web vulnerability. In *Proceedings of 6th Computational Methods in Systems and Software 2022*, *2023*(1), 680–691.

Liu, Z., Fang, Y., Huang, C., & Xu, Y. (2023). MFXSS: An effective XSS vulnerability detection method in JavaScript based on multi-feature model. *Computers & Security*, *124*, 103015.

Mitropoulos, D., Louridas, P., Polychronakis, M., & Keromytis, A. D. (2017). Defending against web application attacks: Approaches, challenges and implications. In *IEEE Transactions on Dependable and Secure Computing*, *16*(2), 188–203.

OWASP Projects. (2014). *Web scarab*. Retrieved from: https://www.owasp.org/index.php/Category:OWASP_WebScarab_Project.

Parvez, M., Zavarsky, P., & Khoury, N. (2015). Analysis of effectiveness of black-box web application scanners in detection of stored SQL injection and stored XSS vulnerabilities. In *2015 IEEE 10th International Conference for Internet Technology and Secured Transactions*, 186–191.

PHP Labware. (2023). *htmLawed*. Retrieved from: https://www.bioinformatics.org/phplabware/internal_utilities/htmLawed/

Port Swigger. (2023). *Automated scanning*. Retrieved from: https://support.portswigger.net/customer/portal/articles/1783127-using-burp-scanner

Rocha, T. S., & Souto, E. (2014). ETSS Detector: A tool to automatically detect cross-site scripting vulnerabilities. In *2014 IEEE 13th International Symposium on Network Computing and Applications*, 306–309.

Saxena, P., Hanna, S., Poosankam, P., & Song, D. (2010). FLAX: Systematic discovery of client-side validation vulnerabilities in rich web applications. In *NDSS Symposium 2010*.

Sharma, S., & Yadav, N. S. (2021). Ensemble-based machine learning techniques for attack detection. In *2021 IEEE 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)*, 1–6.

Sivakorn, S., Polakis, I., & Keromytis, A. D. (2016). The cracked cookie jar: HTTP cookie hijacking and the exposure of private information. In *2016 IEEE Symposium on Security and Privacy*, 724–742.

Snow, K. Z., Rogowski, R., Werner, J., Koo, H., Monrose, F., & Polychronakis, M. (2016). Return to the zombie gadgets: Undermining destructive code reads via code inference attacks. In *2016 IEEE Symposium on Security and Privacy*, 954–968.

Snyder, C. (2019). *What is cross-site scripting (XSS) & how to prevent it*. Retrieved from: https://www.extrahop.com/company/blog/2019/what-is-cross-site-scripting-and-how-to-prevent-xss/

Sutton, M., Greene, A., & Amini, P. (2007). *Fuzzing: Brute force vulnerability discovery*. UK: Pearson.

Tang, P., Qiu, W., Huang, Z., Lian, H., & Liu, G. (2020). Detection of SQL injection based on artificial neural network. *Knowledge-Based Systems*, *190*, 105528.

Tian, W., Yang, J. F., Xu, J., & Si, G. N. (2012). Attack model-based penetration test for SQL injection vulnerability. In *2012 IEEE 36th Annual Computer Software and Applications Conference Workshops*, 589–594.

Trickel, E., Pagani, F., Zhu, C., Dresel, L., Vigna, G., Kruegel, C., . . . , & Doupé, A. (2022). Toss a fault to your Witcher: Applying grey-box coverage-guided mutational fuzzing to detect SQL and command injection vulnerabilities. In *2023 IEEE Symposium on Security and Privacy*, 2658–2675.

Vogt, P., Nentwich, F., Jovanovic, N., Kirda, E., Kruegel, C., & Vigna, G. (2007). Cross site scripting prevention with dynamic data tainting and static analysis. In *NDSS* 2007, *12*.

Wang, R., Xu, G., Zeng, X., Li, X., & Feng, Z. (2018). TT-XSS: A novel taint tracking based dynamic detection framework for DOM cross-site scripting. *Journal of Parallel and Distributed Computing*, *118*, 100–106.

Xie, Y., & Aiken, A. (2006). Static detection of security vulnerabilities in scripting languages. In *USENIX Security Symposium*, 15, 179–192.

Zukran, B., & Siraj, M. M. (2021). Performance comparison on SQL injection and XSS detection using open source vulnerability scanners. In *2021 IEEE International Conference on Data Science and Its Applications*, 61–65.

How to Cite: Sharma, S. (2023). A Study of Vulnerability Scanners for Detecting SQL Injection and XSS Attack in Websites. *Artificial Intelligence and Applications*, *1*(4), 214–220, https://doi.org/10.47852/bonviewAIA3202754