DEVELOPMENT OF A WEB APPLICATION VULNERABILITY SCANNER

ABSTRACT

Web applications have become essential to daily life in the digital age, allowing everything from social connections to e-commerce. They also provide a breeding environment for security flaws like Cross-Site Scripting (XSS) and clickjacking, which let bad actors compromise user data and privacy. Despite these well-known risks, attempts to resolve these problems have been hampered by the lack of an effective and user-friendly web application vulnerability scanner.

This article introduces "TOKY," a specific web application vulnerability scanner designed to find and fix vulnerabilities related to clickjacking and cross-site scripting (XSS). These dangers represent serious concerns to user privacy and data security in the era of pervasive web applications. The study emphasizes the urgent requirement for an effective and user-friendly scanner that addresses the flaws of current solutions. The development process for TOKY is described, along with routines for library usage and detection. Performance analyses show good outcomes in terms of accuracy and rates of vulnerability discovery. A safer digital environment is made possible by TOKY's impact, which goes beyond web development and security to include larger implications for data security and cybersecurity.

I. INTRODUCTION

Web applications have become a crucial co mponent of our daily life in the current dig ital era.
These tools allow us to carry out a variety of tasks quickly and conveniently, from so cial networking platforms to internet storef ronts.
However, as web applications are used mo re often, bad actors looking to profit from s ecurity flaws or steal data have found them to be tempting targets. Cross-
Site Scripting (XSS) and clickjacking vuln erabilities are two of the most common an d dangerous security risks to web applicati ons.

A security flaw called cross-site scripting (XSS) enables attackers to insert malicious programs into web pages that other users are viewing. The victim's web browser will then be used to run these injected scripts, which could result in a number of security flaws, data theft, and unauthorized access. In contrast, clickjacking is a misleading attack where attackers place transparent elements over real web pages to persuade visitors to click on obscure buttons or links. This may cause unexpected activities to be carried out without the user's knowledge, which may result in financial losses or the disclosure of sensitive information.

Web applications are still prone to XSS and clickjacking vulnerabilities despite the known dangers associated with these security weaknesses. Lack of efficient and user-friendly vulnerability scanners is one of the major obstacles to fixing these issues. Developers and security experts can find and fix security flaws in web applications by using vulnerability scanners. To find vulnerabilities, these scanners use a variety of approaches, including as pattern matching, code analysis, and simulated HTTP requests.

This article's main concern is the lack of a thorough and efficient web application vulnerability scanner that is specifically made to find and fix XSS and clickjacking issues. Such a scanner would be a useful tool for web developers and security professionals to improve the security of

their web apps, safeguard user data, and fend off cyberattacks.

The importance of this study goes beyond web development and security. It has significant ramifications for data security, internet privacy, and the larger subject of cybersecurity. We may make substantial progress in securing the digital environment by creating and deploying a dependable scanner capable of proactively identifying and addressing XSS and clickjacking vulnerabilities.

We present "TOKY," a sophisticated web application vulnerability scanner precisely created to find Cross Site Scripting and Clickjacking vulnerabilities, in response to this constantly changing threat scenario. digital developers, security experts, and organizations devoted to protecting their digital assets from exploitation should consider TOKY as a vital tool in their toolbox.

With TOKY, we embark on a journey to enhance web security by providing a robust and efficient solution for identifying these vulnerabilities within web applications. This scanner was developed as a result of a thorough study of the complex web security landscape and a dedication to producing a solution that enables people and companies to actively safeguard their digital assets.

The rest of the paper is organized as follows. Section II describes the literature review. Section III presents the architecture for the proposed system. In Section IV, the implementation and results are discussed. The paper concludes in Section V.

II. LITERATURE REVIEW

Christopher *et al,* (2015) demonstrates how easy it is for attackers to automatically discover and exploit application-level vulnerabilities in a large number of web applications. To this end, we developed SecuBat, a generic and modular web vulnerability scanner that, similar to a port scanner, automatically analyses web sites with the aim of finding exploitable SQL injection and XSS vulnerabilities. Using SecuBat, we were able to find many potentially vulnerable web sites. To verify the accuracy of SecuBat, we picked one hundred interesting web sites from the potential victim list for further analysis and confirmed exploitable flaws in the identified web pages. Among our victims were well-known global companies and a finance ministry. Of course, we notified the administrators of vulnerable sites about potential security problems. More than fifty responded to request additional information or to report that the security hole was closed.

The main contribution of this project is to show how easy it is for attackers to automatically discover and exploit application-level vulnerabilities in a large number of web applications. To this end, we presented SecuBat, a generic and modular web vulnerability scanner that analyzes web sites for exploitable SQL and XSS vulnerabilities. We used SecuBat to identify a large number of potentially vulnerable web sites.

Avanish *et al*, (2012) presented the practical approach of the implementation, for implementing to this approach we have used JavaScript as a programming language and MYSQL as database for storing the data. This consist of three things which are scanner implementation, Attack implementation, and network implementation. It purposes a network-based vulnerability scanner which is able to detect all the pages in a web application which are vulnerable to SQL inject, on behalf of the simulation attack, this tool makes a report which helps programmers to work and fix the vulnerable pages, so this approach helps programmer to focus only on the bad pages rather than the whole web application.

Gurjot *et al,* (2016) stated that according to Gartner Security, the

application layer currently contains 90% of all vulnerabilities. The following list discloses some of the most common attacks that can be performed.

a) **SQL injection:** This is a type of attack that help application's vulnerability that interacts with a database by injecting unauthorized SQL query by an attacker in order to deal its security.
b) **Cross-Site Scripting:** It ensues when an attacker is capable of injecting a script, usually JavaScript, into the output of a web application in such a way that it is executed in the client browser.
c) **Invalidated Input:** Invalidated input refers to data or information that has not been properly validated or checked for accuracy, integrity, or security before being used or processed by a software application.
d) **Broken Access Control:** s a security vulnerability that occurs when an application or system does not properly enforce restrictions on what users are allowed to do. Access control is a fundamental aspect of ensuring the security and privacy of information in a software application or system.
e) **Broken Authentication and Sessions Management:** Broken Authentication and Session Management is a security vulnerability that occurs when an application's authentication and session management mechanisms are not properly implemented or configured.

Danish *et al,* (2012) provide a definition of Web Application Firewalls and their shortcomings and justify the usage of Run-time Application Self Protection. Web Application Firewall intercepts' requests to a potentially vulnerable web application applying rules to evaluate whether a request contains input that might exploit the application. This process requires complex configuration and it may fail open under high load, leaving web applications vulnerable. For a Firewall to function at its peak, there need to know what the vulnerable inputs to the web application are so you can apply the appropriate protections to this input.

Run-time application self-protection stands above any traditional Web Application Firewall, by protecting web applications out of the box with minimal (if any)

configuration needed. This feature could substantially reduce risk by enabling application protection immediately upon deployment. It's capability to instrument at the Application Programming Interface layer allows it to detect attacks precisely.

Sbihi *et al,* (2017) stated that performance evaluation involves measuring various aspects of a vulnerability scanner's operation, including its speed, resource utilization, and accuracy. It helps determine how well the scanner performs in detecting vulnerabilities while considering the impact on the target system and the overall scanning process.

One important metric for performance evaluation is scan time, which measures the duration required for the vulnerability scanner to complete a scan of a web application. A shorter scan time is desirable as it minimizes the time window during which vulnerabilities can be exploited. However, it is important to strike a balance between speed and accuracy, as a faster scan may sacrifice the thoroughness of vulnerability detection.

Idrissi *et al,* (2017) presented that accuracy metrics, such as false positives and false negatives, measure the precision and completeness of vulnerability detection. False positives occur when the scanner incorrectly identifies a vulnerability that does not exist, while false negatives occur when the scanner fails to detect an actual vulnerability. Striving for a low false positive rate helps avoid unnecessary alarms and wasted effort, while a low false negative rate ensures that vulnerabilities are not overlooked.

Additional performance metrics may include the number of vulnerabilities detected, the severity levels assigned to vulnerabilities, and the ability to prioritize vulnerabilities based on their impact and potential risk.

Abdul *et al,* (2014) propose a method of detecting and classifying web application attacks. In contrast to current signature-based security techniques, our solution is an ontology-based approach that specifies web application attacks using the context of consequence, semantic rules and specifications of application protocols.

The system is capable of detecting sophisticated attacks effectively and efficiently by analyzing the specified portion of user requests where attacks are possible. The semantic rules allow us to capture the context of the application, the attack and the protocol that was used. These rules are also utilized for reasoning by inference upon ontological models in order to detect complex polymorphic variations of web application attacks.

•Our semantic approach is capable of representing a shared understanding of structured information about the concepts within a specific domain, provides reasoning features and the ability to automatically analyze information. It also specifies semantic relationships between concepts, improves interoperability and encourages reuse and evolution over time.

Most of the tradition security solutions such as MoD Security [59] and Snort [60], lack an inference capability; that is why these systems depend upon attack signatures (rules) for each header and request body where parameters are present. Moreover, they have to search sequentially to locate parameters that are present. This can mean looking at each header and request body. Thus, they require more attack signatures and it often takes more time to search for an attack vector. Whereas our approach with its set of rules and inference capability overcomes these limitations. The mechanism uses inference rules and reduces the search space via a simultaneous search of the parameters from headers and the request body. It then applies a single semantic rule on these parameters for validation.

We present examples and results which demonstrate that our semantic approach can be used to detect zero day and more sophisticated attacks in real-world programs.

III. METHODOLODGY

The proposed scanner is developed through four unique stages, each of which contributes to a final goal and the operational functioning of the scanner.

a) Libraries Importation: the libraries used for the development of this scanner is requests which allows the script to make HTTP requests to the specified URL and re which provides supports for regular expression. Regular expressions are used for pattern matching, which is crucial for detecting XSS vulnerabilities.
b) XSS detection function: performs a Cross-Site Scripting (XSS) vulnerability analysis on a supplied URL. It does this by first utilizing the requests.get() method to send a GET request to the specified URL. The server response is then examined for potential XSS attack vectors using a collection of predefined XSS patterns created using regular expressions. To store any discovered matches, the method creates an empty list called xss_matches. Using re.findall(), it loops through the list of patterns to find every instance of a match in the response text. The xss_matches list is expanded if any matches are discovered. The function then produces a list of all found XSS matches.
c) Clickjacking detection function: examines a URL for possible

Clickjacking weaknesses. It takes a URL as input to get started. Then, it starts a GET request using requests to the supplied URL. Get(). The X-Frame-Options header is then checked in the response for its presence and specified value. This header indicates a potential susceptibility to Clickjacking if it is either empty or set to 'DENY'. The function ultimately produces a Boolean value that denotes whether or not a Clickjacking vulnerability exists.

d) Main function: coordinates the scanner's operations. It acts as the main location for execution. It asks the user to enter a target URL to be evaluated at first. Following that, it calls two crucial functions: detect_xss() to find Cross-Site Scripting flaws and detect_clickjacking() to find Clickjacking flaws. The outcomes of these functions are then organized into a results_data dictionary. This information is then logged using a with statement and the open() function into a file called scan_results.txt. The target URL, any discovered XSS vulnerabilities, and the state of Clickjacking are all documented in the file. The user is then given feedback indicating that the scanning procedure is finished.

IV. EVALUATION

The hardware and software requirements of the system are as follows:

Operating system:	Cross Platform
RAM:	2GB or greater
Processor Speed:	1.8GHz or greater
Processor:	Dual Core or greater
Python version:	3x

The performance evaluation metrics for the web application vulnerability scanner were performed based on Accuracy and vulnerability detection rate (VDR). The Table 1 shows the results evaluation.

1. Accuracy for XSS vulnerability is calculated by

   i. Accuracy for XSS = $\frac{TP + TN}{NI}$ = 0.95 =95%

Accuracy for Clickjacking vulnerability is calculated by

   ii. Accuracy for Clickjacking $= \frac{TP + TN}{NI} = 0.85 = 85\%$

2. Vulnerability detection rate (VDR) is calculated as VDR $= \frac{NP}{NP+FN} = 0.947 = 95.5\%$

V. CONCLUSION

In conclusion, the development of web applications has unquestionably transformed how we engage with the digital world. However, tremendous convenience also comes with great responsibility, and in this case, that responsibility is to make sure that these applications are secure and reliable. Our experience with "TOKY," a dedicated web application vulnerability scanner, during its development and review highlights the crucial relevance of preventative security measures.

Cross-Site Scripting (XSS) and clickjacking vulnerabilities continue to exist, serving as a sharp reminder that risks exist in the digital sphere as well. Despite the acknowledged risks connected to these security flaws, our ability to effectively tackle them has been hampered by the lack of an effective and user-friendly vulnerability scanner.

TOKY represents a substantial advancement in web application security because to its comprehensive detection capabilities and reliable methods. It provides enterprises with a potent weapon to protect their digital assets and gives a glimpse of hope in the ongoing fight against cyber attacks.

VI. REFERENCES

[1] Stefan K, Engin K, Christopher k, and Nenad (2015). Secubat: Web application

vulnerability scanner. https://www.studocu.com/in/document/savitribai-phule-pune-university/information-and-cyber-security/www2006-secubat-z-cs/53611553

[2] Avinash. K.S (2012) A Network Based Vulnerability Scanner for Detecting SQL Attacks in Web Application. https://www.researchgate.net/publication/254031416_A_network_based_vulnerability_scanner_for_detecting_SQLI_attacks_in_web_applications

[3] Gurjot S, Rajdeep K, and Arshdeep K (2016) "Distinctive Vulnerabilities in Web Applications" *International Journal of Computer Science and information technology, 4(2) 35-42*

[4] S. El I.Berbiche N.Guerouate F and sbihiM. (2017) "Performance Evaluation Of Web Application Security Scanners For Prevention and Protection Against Vulnerability" International Journal of Applied Engineering Research ISSN 0973-4562 Vol 12, No 21

[5] Abdul R, Khalid L, Farooq A, Ali H, Zahid A (2014) "Semantic Security Against Web Application Attacks" *International Journal of Information Science vol. 254, pp 19-38*