WVS: The Threat Scanner

Piyush
Computer science and engineering, Galgotias University greater noida, India
piyush.scsebtech@galgotiasunivers ity.edu.in

Nikhil kaushik
Computer Science and Engineering, Galgotias University Greater Noida,India
nikhil_kaushik.scsebtech@galgotia suniversity.edu.in

Abstract—

Currently the SDLC should be incorporate the security feature. Attacks like Input Validation Attack which are the most diverse attacks on the web applications. Our main intention is to be focused on the detection and prevention of Input Validation attacks like Cross Site Scripting cross site scripting (XSS), SQL Injection, DDOS etc. These are types of attacks on web pages and account as the unsafe vulnerabilities existed in the web applications. Once the vulnerabilities are detected, the hacker can easily get access to the aunthecated data of the web application and get access to the database of the web application. The hacker may perform various attacks like session-hijacking, cookie stealing, or various other malicious attacks. To prevent this type of attacks it is very important for user to implement the security measures that can easily blocks the web browser attacks. GET and POST are 2 methods which are responsible for the exploit of vulnerabilities over network through web requests. In this project, we are focusing on detection and information about preventing the attacks. The unavailability and high price of vulnerability scanners for detecting vulnerabilities in web applications which leads to hijacking and stealing the data from server which creates a security problem for the organization. So, making of an affordable scanner is very important.
In this project we are going to develop vulnerability scanner for finding vulnerabilities in web application and provides the information about its remediations.

Keywords—Sql injection, Cross site scripting, remediation, vulnerability, hijacking

I. INTRODUCTION

As of January 2022, there have been over 1.97 billion websites on the web. Hackers attack every 36 seconds, on the average 2,344 times each day. This gives us the idea that many websites on the Internet are vulnerable to different attacks. As of the end of 2021, 46% of publicly facing websites are prone to SQL Injection and 18% to Cross Site Scripting attacks [3]. Another such attack, in August 2019, was on the famous coffee chain 'Starbucks' web services that created a way to access their critical database through the SQL Injection Vulnerability. From above discussion, we can say that Security plays an important role in developing websites.[9] but unfortunately, the web developers are not aware of these type of security attacks it resulting in more vulnerable websites which cause lots of security attacks. Some of the most common attacks are SQL injection, XSS, So we are developing a system that will find these vulnerabilities in given web applications and report them to the user of the system. We are developing a system that will takes site URL as input from the user[8]. The system will scan the target URL in an Automated way. Then it will scan all collected URLs and it will test different payloads to exploit all the vulnerabilities. Finally, scanner will generate a report which contains the detected vulnerabilities and payloads used.

Currently the software development life cycle should be incorporate the security feature. Attacks like Input Validation Attack which are the most diverse attacks on the web applications. Our main intention is to be focused on the detection and prevention of Input Validation attacks like Cross Site Scripting cross site scripting (XSS), SQL Injection, These are types of attacks on web pages and account as the unsafe vulnerabilities existed in the web applications. Once the vulnerabilities are detected, the hacker advances intend is to access of the authenticate user's web- browser and may perform various attacks like session-hijacking, cookie stealing, or various other malicious attacks. To prevent this type of attacks it is very important for user to implement the security measures that can easily blocks the web browser attacks. GET and POST are 2 methods which are responsible for the exploit of vulnerabilities over network through web requests. In this project, we are focusing on injection, detection and information about preventing these attacks. The unavailability and high price of vulnerability scanners for detecting vulnerabilities in web applications which leads to hijacking and stealing the data from server which creates a security problem for the organization as well as government people.

so, making of an affordable scanner is very important. In this project we are going to develop vulnerability scanner for finding vulnerabilities in web application and provide the information about its remediation.

Complete web vulnerability scanner is used to find the website bug. and after finding the bugs it will shows the types of bugs in that website. Later these bugs are prevented. This project is developed in python.

As we know that increase in number of data breaches harms the organization in past years. Most common attack that is attacked on a website by a hacker is "Injection Attacks" In injection attacks a maliciouscode is injected in the website.

Increasing dependency on the modern technology advanced forced on network and information system. It creates exploitable vulnerabilities which can be easily This paper is structured as follow. Section 2 explain about the existing works. Section 3 exaplain about

different types of Vulnerabilities.
Section IV containthe proposed system and system architecture
In Section V – it will tell us about the specification ofthe system. In Section VI, It contains conclusion and future scope of the project.

II. LITERATURE SURVEY

An entire web application security testing solution that will be used both standalone and as a part ofcomplex environments. [3] It offers built-in vulnerability assessment and vulnerability management, also as many options for integration withmarket-leading software development tools. It is not an open-source tool. It is the most expensive tool available. Burp Suite Web Vulnerability Scanner: Burp Scanner used the Port Swigger the world's most leading research to assist its users to hunt out an honest range of vulnerabilities in web applications, automatically. [5]. Web application scanner is the dynamic deep scanning which covers all the apps on your perimeters, in the internal environments and the under active development, even APIs that support the mobile devices. [4]. Nessus is the best vulnerabilities assessments solution for the security practitioners. The latest update of this software is very easy to use but is is also expensive one.[9] The Case of Cross-Site it Request falsification published within the year 2020 by Stefano, Alvise Rabi, Mauro Coti, Ricardo Focrdi, , Gabriele Tolomei. It has the best key advantages of the offering a language-unbeliever vulnerability.
The detection approaches, which are abstracts from thecomplexity of the scripting languages and offers a consistent interface to widest possible range of the web application. [10] this is an structured algorithm and tool that will detect the dangerous websites vulnerabilities in the year 2022 and written by Hoag Vit ong, Tog Ah Tuan, David Tiar. The new technique which are proposed has the advantage of detecting the attacks in the nested SQL queries and giving the good performance. [12] The Automated Vulnerability Scanning tool with Multiple Vulnerabilities detection within the year 2019 published by Xun Zang, Jiiong Zho, Fan Yang, Qin Zhag, Zhiu Li, Xujn Zhag. [15].
An Vulnerability Scanning on Machine Learning in the year 2019 by Xiang TIAN, Di TANG, established the standardized data sets for the different industries and the different businesses which help to improve the quality of testing. [16] Commix: in 2019 by Anasios Staopoulos, Chroros Ntagian and Christos Xeis it exploits the command like vulnerabilities in websites. It supports the excess of functionalities that can attempts to cover the different authentications mechanism, custom header, tornet working, attack vectors and the various exploitation scenarios produced by the programming languages, system user inventory. [6] Dimtris R. Sios, Jovan Zivic , they proposed the Automated Combinatorial Testing for Detecting theSQL Vulnerabilities in the Web Applications in year 2019. It demonstrates that our approach can be successfully tackle in faulty filtering mechanisms. [8]

III. WEB SECURITY VULNERABILITIES:

A. SQL Injection:

SQL injection attack are one of the tops most threats in database centric web application and the SQL injectionvulnerabilities are the foremost serious Vulnerabilities types. SQL Injection

allow the attacker to gain the control over the database of an application. [7] Every other website needs input from the user for a variety of reasons and if they are not validated properly, theymight lead to some critical issue. Consider the login function where user has to provide a username and password. These credentials then validated at the backend through the SQL query statements and if theyare correct, then the user is successfully logged in.
Now let's consider a situation where it can be abused. If the user provides some malformed inputs and the application accepts it as it is, then the attacker can leverage this to perform a database attack. From Fig. 2 we can see that the attacker Alice is providing username as 'admin" ;– ' and some arbitrary password.

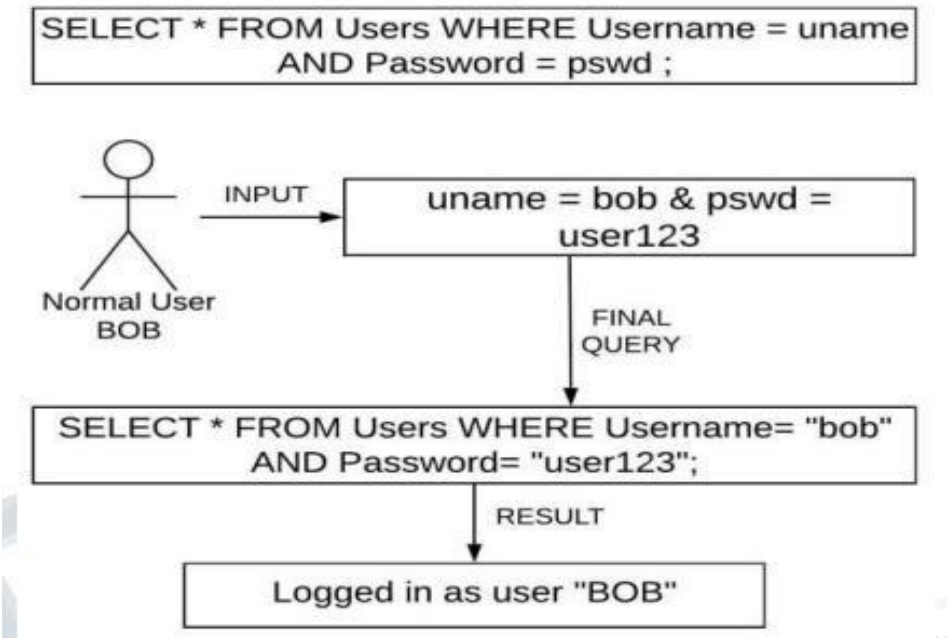
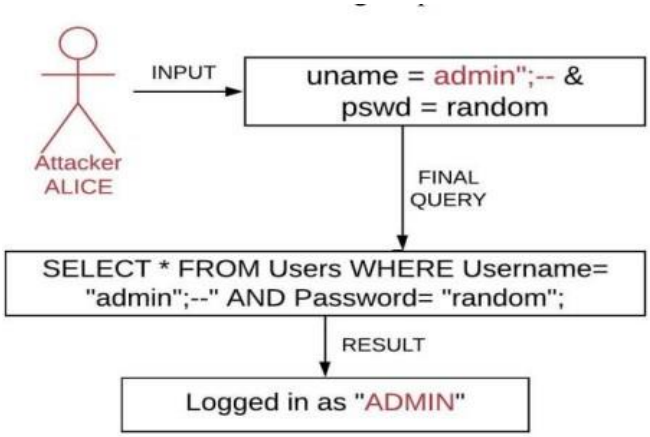


Fig 1 SQL injection User prospective

This results in breaking of the structure of the SQL query used at the backend. So the effective query will be'SELECT*FROMUsersWHERE
username="admin";– " AND Password="random";'. This will to lead to a change in the application logic, as the double-quote entered in the username will match thestarting doublequote of the query and as the '–' is considered as an identifier for comment in most of the relational databases, it simply comments out the succeeding part of the query. So the new query will be 'SELECT * FROM Users WHERE username="admin";'.
The attacker can now log in to an account withoutknowing the password.

Fig 2 SQL injection Attackers prospective



B . Cross Site Scripting :

The Cross-Site Scripting attack is one of the critical vulnerabilities that can affects the web applications security. In This attack the malicious scripted code injects into the web application by the attacker in the client-side within the users browser or in the server side within the database, this malicious script is written in the JavaScript code and injected within distrustful input data on the web applications [8]. Many applications which provides the facilities to search for particular content. Whenever the user searches the required content, the relevant result is displayed on the webpage along with the search keywords which is entered by the user.

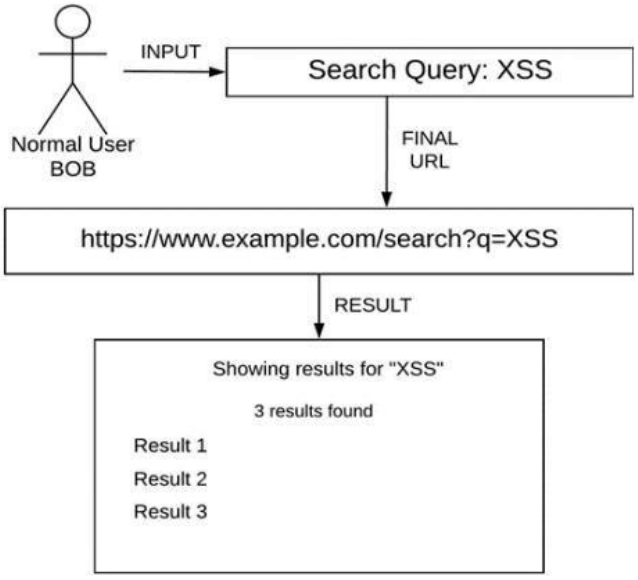


Fig 3 cross site scripting User prospective

Now let us consider the attacker's perspective. The malicious user makes use of this search functionality as any of the normal user and checks whether the searched keyword gets reflected on the resultant page returned by the application. If it succeeds, then the attacker comes to know that there is a possibility of XSS to take place at that particular location. If

the application does not perform either encoding or filtering on the search query given by the user, then it might be possible for an attacker to break out of the previous HTML tag and insert a new one. From the figure given below, we can see that the attacker is able to insert a new script tagthat can be used for malicious purposes. Being able to insert a new script tag can have several consequences, including but not limited to, stealing session cookies, bypassing CSRF protections, theft of user's. personal and sensitive data. In some extreme scenarios, it might be possible to exploit the user browser by leveraging the cross sote scripting.

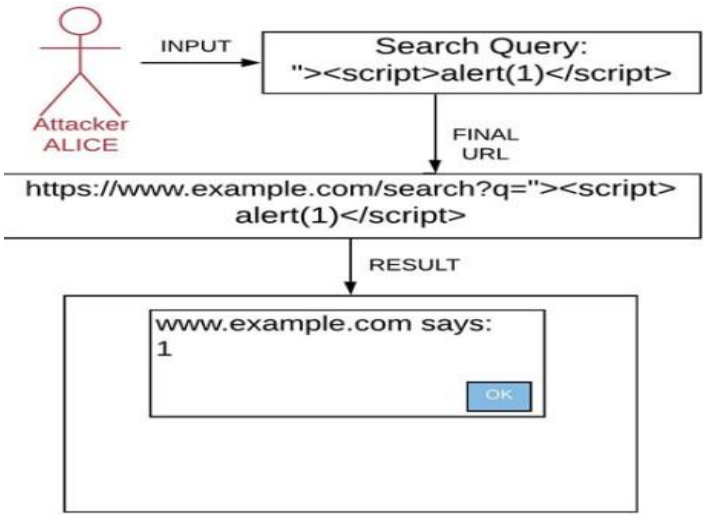


Fig 4 cross site scripting Attacker prospective

IV. PROPOSED SYSTEM

A. System Architecture

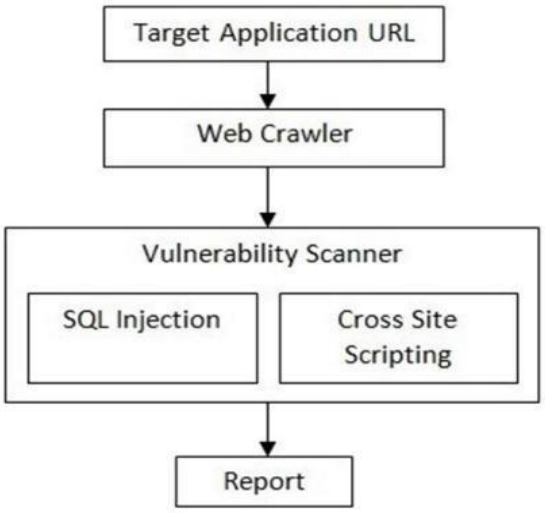


Fig 5 system architecture

In the application user going to gave application URL as a input after that application going to scan that URL and scan the vulneribilities like SQL injection Cross Site Scripting also going to scan other vulnerabilities. it will detect the malicious code in the website which tell user about the vulneribilites present in the wesite or web applications.

Project Scope

Main scope of project is to provide an organization to scan their website at lower and affordable price with better result. The system will scan the target application and check if the web application is having any of the vulnerabilities: After scanning it will show all the remedies to the user which vulnerability is present in website.

B. User classes and characteristics

- **WebSpider: –**

Robots.txt Parsing

∗ Checks for the presence of the robots.txt file and if present, collect allowed and disallowed URLs.

– URL Parsing:

∗ All URLs specified within the anchor tag from the current page are saved in a List.

∗ Multiple threads will be created to crawldifferent hyperlinks simultaneously.

∗ Relative URLs (like /admin or #footer) are converted into Absolute URL (like https://example.com/admin orhttps://example.com#footer)

∗ URLs which are not in the scope of target application are removed from the list (for example twitter.com or instagram.com)

∗ Hyperlinks with ‘mailto:’ or ‘javascript:’ andthose pointing to static file types like images, pdfs, fonts, etc. are also removed.

- **Web Scanner:**

– Search for form elements in crawled URLs. From the listed form elements, find out input fields. – Pass the appropriate payloads to the input field and save the response received from the server.

- SQL Injection:

For an Error-Based SQL Injection attack, we try to break the syntax of the SQL query being used by the server by passing SQL-special characters (E.g.’,”, etc.) through user input.

For Union SQL injections, a dataset consisting of SQL queries of specific types will be created. – In the user input, these payloads will be passed to the server to checkif SQL query is well formed after inserting the given payload. The system requires the target URL to be entered by the user. If the response from the server for the payload is similar to the usual response, then we can infer that SQL injection is possible.

Cross Site Scripting:

For Cross Site Scripting, a dataset will be created consisting of different XSS (cross site scripting) payloads. These payloads will then be tested against all the input fields present on that web page The responses are checked for the presence of a particular payload and to check if XSS is successful. If successful, that part of the web page is considered vulnerable and a report will be generated giving a detailed knowledge about the vulnerability detected.

V. OTHER SPECIFICATIONS

A. Advantages

- Supports automated and reliable Scanning.
- Optimized use of the number of threads to control the load on the target application.
- Fully Detailed vulnerability analysis.
- User-friendly GUI.
- Work in linux.

B. Applications

- Identifying and reporting the vulnerabilities present in the web application.

C. CONCLUSION AND FUTURE SCOPE

We have done scanning and find some of the common vulnerabilities on the websites , such as SQL Injection and the XSS. We have proposed an system which can find more vulnerabilities produce further enhancements in the system to improve the ability of the vulnerabilities detection in the websites. We proposed a system that will scan the entire web application, find different types of vulnerabilities, and generate a report which contain all the information about the founded vulnerabilities. This system uses various tools to find the vulnerabilities in the websites correctly. There is a future scope of improvement in various aspects of the developed system. The time required for scanning can be improved so that more complex applications can also be tested for vulnerabilities which took less time to scan the websites.

Acknowledgment We thank Ms Urvashi Sugandh, Assitant Professor, Department of Computer science and Engineering, for giving us proper guidance for our final year project, all With respect and gratitude, we would like to thanks all the peoples, who have helped us directly or indirectly in our project.

C. REFERENCES

[1] Stefano Calzavara, Mauro Conti, Riccardo Focardi, Alvise Rabitti, and Gabriele Tolomei. Machine learning for web vulnerability detection: The case of cross-site request forgery. IEEE Security & Privacy, 18(3):8–16, 2020.

[2] Hoang Viet Long, Tong Anh Tuan, David Taniar, Nguyen Van Can, Hoang Minh Hue, and Nguyen Thi Kim Son. An efficient algorithm and tool for detecting dangerous website vulnerabilities. International Journal of Web and Grid Services, 16(1):81–104, 2020.

[3] S. K. Mahmoud, M. Alfonse, M. I. Roushdy, and A. M. Salem. A comparative analysis of cross site scripting (xss) detecting and defensive techniques. In 2017 Eighth International Conference on Intelligent Computing and Information Systems (ICICIS), pages 36–42, 2017.

[4] C. Sharma and S. C. Jain. Analysis and classification of sql injection vulnerabilities and attacks on web applications. In 2014 International Conference on Advances in Engineering Technology Research (ICAETR - 2014), pages 1– 6, 2014.

[5] Dimitris E Simos, Jovan Zivanovic, and Manuel Leithner. Automated combinatorial testing for detecting sql vulnerabilities in web applications. In 2019 IEEE/ACM 14th International Workshop on Automation of Software Test (AST), pages 55–61. IEEE, 2019.

[6] Anastasios Stasinopoulos, Christoforos Ntantogian, and Christos Xenakis. Commix: automating evaluation and exploitation of command injection vulnerabilities in web applications. International Journal of Information Security, 18(1):49–72, 2019.

[7] TIAN Xiaopeng and TANG Di. A distributed vulnerability scanning on machine learning. In 2019 6th International Conference on Information Science and Control Engineering (ICISCE), pages 32–35. IEEE, 2019.

[8] Xun Zhang, Jinxiong Zhao, Fan Yang, Qin Zhang, Zhiru Li, Bo Gong, Yong Zhi, and Xuejun Zhang. An automated composite scanning tool with multiple vulnerabilities. In 2019 IEEE 3rdAdvanced Information Management, Communicates, Electronic and AutomationControl Conference (IMCEC), pages 1060–1064. IEEE, 2019.

[9] August 2019 International Journal on Advances in Internet Technology 12(1 and 2):12-27 DOI:10.21256/zhaw-17956

[10] Fan Yang, Qin Zhang, Zhiru Li, Bo Gong, Yong Zhi, and Xuejun Zhang. An automated composite scanning tool with multiple vulnerabilities. In 2019 IEEE 3rd Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), pages 1060–1064. IEEE, 2019.

[11] N. Antunes and M. Vieira, ‘‘Defending against web application vulnerabilities,’’ Computer, vol. 45,no. 2, pp. 66–72, Feb. 2012, doi: 10.1109/MC.2011.259.

[12] N. Antunes and M. Vieira, ‘‘Designingvulnerability testing tools for web services: Approach, components, and tools,’’ Int. J.Inf. Secur., vol. 16, no. 4, pp. 435–457, Jun. 2016, doi: 10.1007/s10207-016-0334-0.

[13] L. Auronen, ‘‘Tool-based approach to assessingweb application security,’’ Helsinki Univ. Technol., Espoo, Finland, Tech. Rep.T-110.501, 2002, vol. 11, pp.12–13. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.104.893&rep=rep1&type=pdf

[14] N. F. Awang and A. A. Manaf, ‘‘Detecting vulnerabilities in web applications using automated black box and manual penetration testing,’’in Proc. Int.Conf. Secur. Inf. Commun. Netw. Cairo, Egypt: Springer, Sep. 2013, pp. 230–239, doi: 10.1007/978-3-642-40597-6_20.

[15] T. Basso, P. C. S. Fernandes, M. Jino, and R. Moraes, ‘‘Analysis of the effect of Java software faultson security vulnerabilities and their detection by commercial web vulnerability scanner tool,’’ in Proc. Int. Conf. Dependable Syst. Netw. Workshops (DSN- W), Jun 2010, pp. 150–155, doi:10.1109/DSNW.2010.5542602.

[16] J. Bau, E. Bursztein, D. Gupta, and J. Mitchell, ‘‘State of the art: Automated black-box web application vulnerability testing,’’ in Proc. IEEE Symp. Secur. Privacy, May 2010, pp. 332–345, doi: 10.1145/1135777.1135817.