Volume 9, Issue 5, May – 2024 International Journal of Innovative Science and Research Technology
ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24MAY501

Understanding SQL Injection Attacks: Best Practices for Web Application Security

Tanzila Hasan Pinky[1]; Kaniz Ferdous[2]; Jarin Tasnim[3]; Kazi Shohaib Islam[4]

Abstract:- SQL (Structured Query Language) injection represents a security weakness that enables attackers to run SQL commands within a web applications database. When exploiting a designed application lacking input validation a malicious actor can control input data to execute SQL queries. The objective of detecting SQL injection vulnerabilities is to identify any section of a web application to user input exploitation, for SQL injection attacks and confirm that the application adequately validates user inputs. The aim of this project is to try and form an attack chain and test the same against any website to assess the website for any weak links and identify any entry points that an attacker could use to penetrate the system and take control of the same.

From the paper it is figured that most of the tools only check the vulnerability for the given URL and do not crawl through the webpages and find if the vulnerability is present in any of the other pages. In this project, we are taking the additional step to confirm that there are no vulnerabilities mentioned in this research present in any of the webpages.

Keywords:- *SQL Injection, SQL Queries, Vulnerabilities, Website, URL, Webpages.*

I. INTRODUCTION

In today's era web applications are essential, in our lives helping us with tasks like online shopping and sharing data. The main goal of this study is two pronged; first to examine how well web applications can withstand security risks and second to suggest and assess methods to improve their ability to defend against threats. t's not only the option to defend ourselves from the attacker but also think like them to check the security of the webpages from the queries of the attackers. By this, all the points of the security will be tested and implemented carefully. For browsing the online pages and make them more user friendly as well trustworthy this research will be more convenient and facilizing for the users.

However, relying on these platforms also exposes us to security risks that could compromise our information. As students studying computer science, it's our duty to protect.

This data and keep the confidentiality of the user's information's. Our study focuses on evaluating the security of web applications specifically looking into vulnerabilities such as SQL injection, Cross Site Scripting (XSS), and HTTP Strict Transport Security (HSTS). Our goal is to pinpoint weaknesses and suggest ways to strengthen defence mechanisms. Divided into three parts our research delves into testing for SQL injection XSS scripting and HSTS vulnerabilities. This study aims to provide a comprehension of web application security paradigms by combining real world data and theoretical perspectives. By examining these weaknesses and exploring ways to mitigate them we aim to offer insights, for improving web application security. Through this study, we aim to enhance understanding of web application security and help stakeholders defend against threats while maintaining user trust in platforms. For the testing environment we are going to use the following structure below:

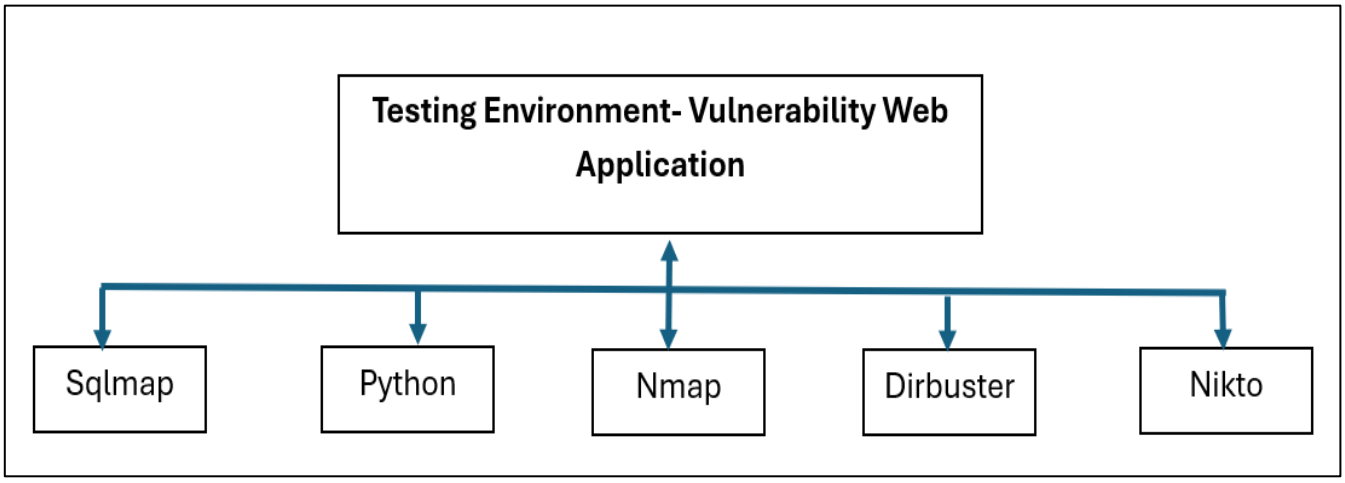


Fig 1: Proposed Architecture of the Testing Applications

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24MAY501

In this paper we will proceed with the steps; (II) presenting the findings of our research and insights gained from them (III) demonstrating the outcomes of our study and research by outlining the challenges for our strategy, (IV) following by a discussion on the significance of our discoveries and (V) potential areas for further exploration, in future work.

II. RELATED WORKS

Research on the use of the black-box method for penetration testing, with a specific focus on SQL injection attacks. This research only gave more statistical information than technical information and goes on to explain the vulnerability of every website and how it is important to perform penetration testing at regular intervals to better safeguard the web application [1]. Oliver Moradov on SQL Injection Testing methods [10] mentioned in a blog, there are 5 types of testing for an SQL Injection: Stacked Query Testing (Completing an SQL query and writing a new one on top of the previous), Error-Based Injection testing (exploiting SQL error messages), Boolean-Based Injection Testing (adding conditional statements), Out-of-Band (Blind) Exploit Testing and Time Delay Exploit Testing (monitoring server response time). The fist documentation for SQL injection was exploited in 1998 by a researcher and hacker where it was mentioned that with basic coding skills, one could use unauthorized SQL commands on legitimate SQL statements to extract confidential information from a database. There are multiple tools to test for every one of these vulnerabilities separately and there are a lot of applications that offer tests for various vulnerabilities combined. But all these tools require basic knowledge of how the application works and the kind of parameters that need to be sent to execute the tests [8]. In another research paper network traffic patterns uses machine learning algorithms to identify unusual or suspicious traffic patterns that may indicate an attempt to launch a SQL injection attack. The proposed detection system employs machine learning methods to identify patterns and scrutinize gathered data for indications of an SQL injection risk [3]. Since this isn't a tool there could be hurdles, in constructing the model and initiating the testing phase. Like the machine learning model, another proposal uses runtime validation to detect and prevent SQL Injection Attacks (SQLIA). The model uses a proxy server as a middle layer between the client and database server to check input queries and filter out any SQLIA [12]. For an understanding of various detection and prevention techniques like Hybridization of Knuth-Morris-Pratt (KPM) and Boyer-Moore (BM) algorithm, detection model to scan SQL injection on the web environment for the SQL Injection attacks like Tautology, Union Queries, Piggy-Backed attack etc, which will be useful for us to write different SQL Injection attack queries [2].The cheat sheet (Invicti, n.d.), like other blogs, gives an in-depth explanation of different SQL injection attacks on MySQL, Microsoft SQL Server, ORACLE and PostgreSQL SQL servers [5]. SQL Injection blog gives a good introduction on how to test for SQL Injection attacks. From this blog, we will be using the examples provided for basic use cases [16]. In another blog, there are given examples for each testing method. Using these queries, we will test for each type of SQL Injection attack [15].Moving from simpler use cases, we will use queries from the exhaustive list provided by the OWASP organization to test for advanced cases for our research testing [13].For a deeper explanation of SQL Injection with the threat modelling, explain the impact of the attack and what the attacker is capable of, applications where SQL Injection is common like PHP and ASP and the severity of SQL Injection attacks. Also, briefing the occurrence and the consequences of the attack. Making this remark makes one wonder about how we could link a SQL Injection attack with other forms of vulnerability [14]. In a blog the major problem arises because there are not enough tests written to validate all negative use cases when the code is developed. Organizations might not have enough funding to dedicate a testing team or have sufficient manpower. For a persistent attacker, this is an easy target. The author suggests using Feedback-Based Fuzzing to overcome the issues. This automates the testing process along with generating the test data based on the previous results [9].

III. PROPOSED MODEL

➢ *System Construction*

This model is a collection of tools used for checking vulnerabilities in a website. This framework uses various open-source applications to identify different types of vulnerabilities. We implement Vulnerability Checker using Python and vulnerability testing software like SQLMap, Nmap, DirBuster, Nikto and various Python modules to test the website for SQL Injection attacks, Man-in-the-Middle attacks by testing if the web application has HTTP Strict-Transport-Policy and Cross-Site scripting attack.

- SQLMap: SQLMap is an open-source penetration testing tool used for testing SQL injections which generates the SQL queries to test against the website. SQLMap has a lot of options to choose from and it is extensive to test for sql injection. Basic arguments required to execute the sqlmap command is '-u' option, to specify we are going to enter an URL and the second argument is the URL itself.
- Nmap: Nmap short for Network mapper, is an open-source application to discover network and audit security vulnerabilities. Nmap needs the URL as a mandatory argument and depending on the requirements additional options can be provided to get the desired output.
- Dir buster: Dir buster is a Java application used for brute forcing files and directories on a web application. It tries to find any hidden files or directories within the application. Dir buster tool reads all files and directories on the website and displays on the terminal. The output can also be redirected to an output file.
- Nikto: Nikto is an open-source application that scans web servers and common gateway interfaces. Nikto provides support for HTTP proxy, accepts cookies, and verifies if the server is running on any outdated components. Nikto takes 2 inputs as arguments. First, -h to indicate host URL and the second argument URL or the IP address of the web application to test.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24MAY501

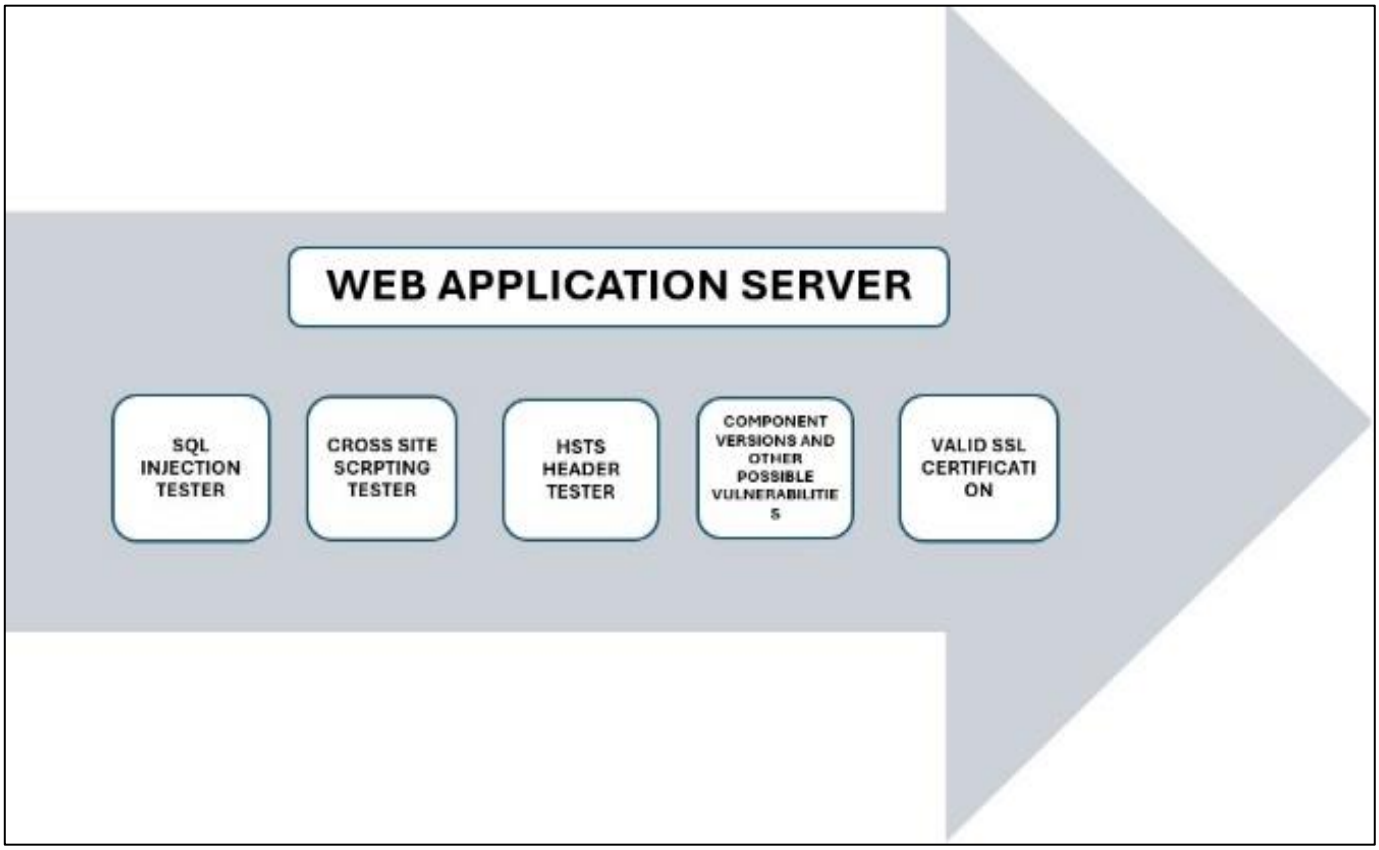


Fig 2: Flow Chat of the System Process

A Python script combines it all and imports sqlmap-dev version. All options that need to be executed are sent as arguments to the sqlmap-dev script and the command is executed with the appropriate values and the result is returned which can be redirected to a file. Various options of sqlmap are tested the same way. Once the results are taken, to check for the strength web application, by checking if the HSTS header is present in the first HTTPS response from the server. If the HSTS header is present, then the website cannot be downgraded to an HTTP site. Moving on to the next level of testing for cross-site scripting possibilities in the web application. Retrieving all the forms available on the webpage and insert a Java script into the forms. If the website executes the malicious script, then the website is vulnerable to cross-site scripting attacks. All these tools check for the URL specified and do not go forward and check if any other web pages on the site is vulnerable to these attacks. Using an application called Dir buster, we can extract all hidden directories and files and try to navigate to these web pages. By passing these webpages as arguments to the script, we can check if all the webpages on the website is vulnerable to any security risks.

➢ *Developed System*

The Python script includes a set of tools that thoroughly check the website's compliance, with cybersecurity standards especially focusing on HTTP Strict Transport Security (HSTS) and the validity of Secure Socket Layer (SSL) certificates. It also evaluates the susceptibility, to cross-site scripting (XSS) by examining web forms. To detect SQL Injection the script uses a subprocess to handle command line inputs. It is designed for users of all skill levels providing two testing options: Intermediate and Expert. The script implements a comprehensive suite of tests, covering the gamut of injection techniques including Error-based, Time-based, and Union Query-based.

- SQL Injection Testing: Sqlmap additionally can be used to automate various tasks, making it a valuable tool for streamlining the testing process and maximizing efficiency. Upon executing the SQL injection test, it was determined that the target object was not susceptible to injection. The testing procedure proceeds to assess the remaining links discovered on the webpage. Further evaluation is performed to ensure that the website remains secure against potential security threats.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24MAY501

Fig 3: Intermediate Level Users Test of SQL Map Test

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24MAY501

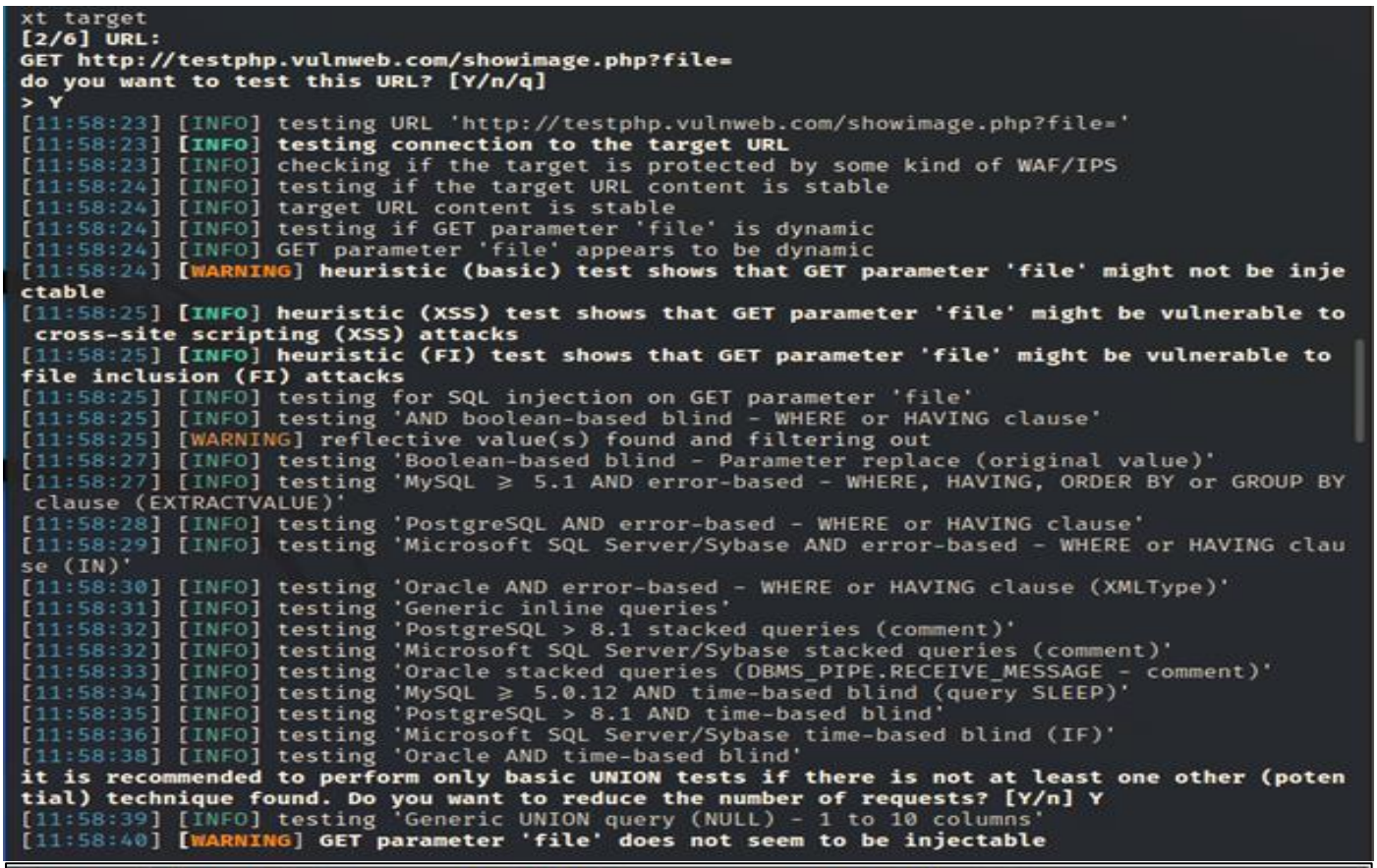


Fig 4: For Second Level of Webpage SQL Injection Test

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24MAY501

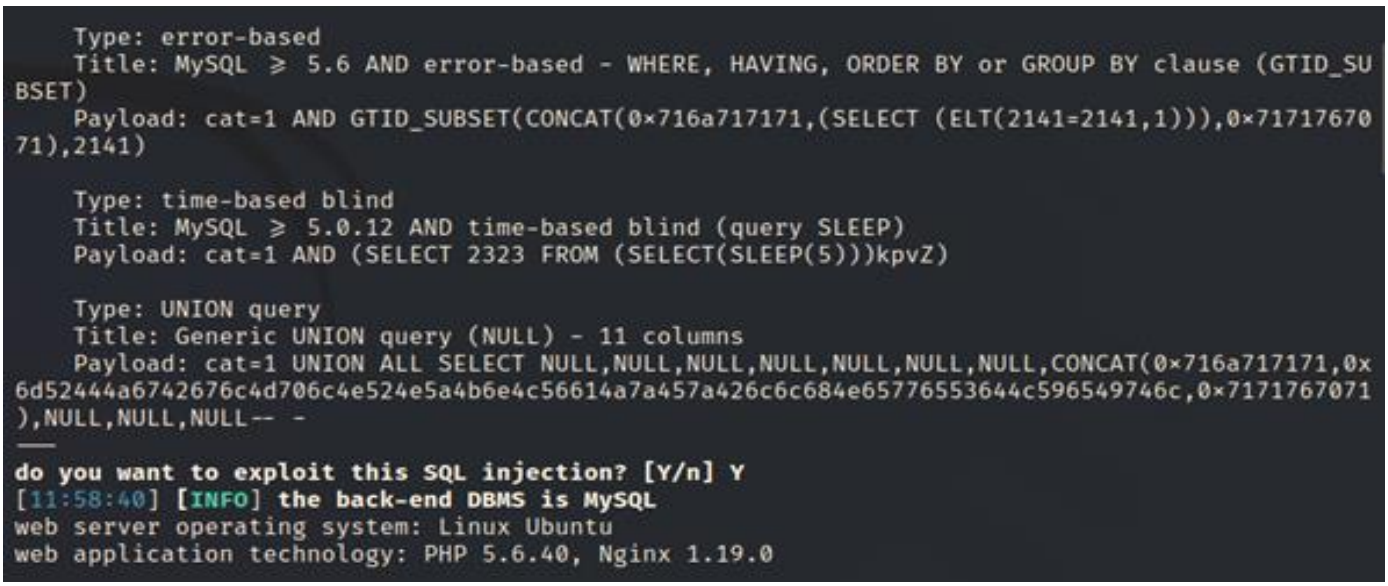


Fig 5: Both of Figures are Test Result for SQL Injection

- Cross-Site Scripting Testing: The cross-site scripting assessments are not limited to the initial URL provided by the user, but rather encompass a broader examination of other pages available on the website to identify those that are susceptible to XSS attacks. For intermediate level users we go a level up and test all available webpages on the website. To retrieve the available webpages, we use dirb to find any hidden files or folders. Custom Word List generator generates a wordlist based on the words present on the website. Dirb takes as argument the wordlists created by CeWL and a predefined wordlist to navigate through the webpages to identify any hidden folder or files. Dirb identifies hidden folders and files. We store these findings in an output file and use the found links to test for presence of cross-site scripting in all these webpages. Cross-site scripting for intermediate level users is done not just for the given website, but also for other files found using dirb tool. The script iterates over the found webpages and applies cross-site scripting tests to identify the vulnerability if any.

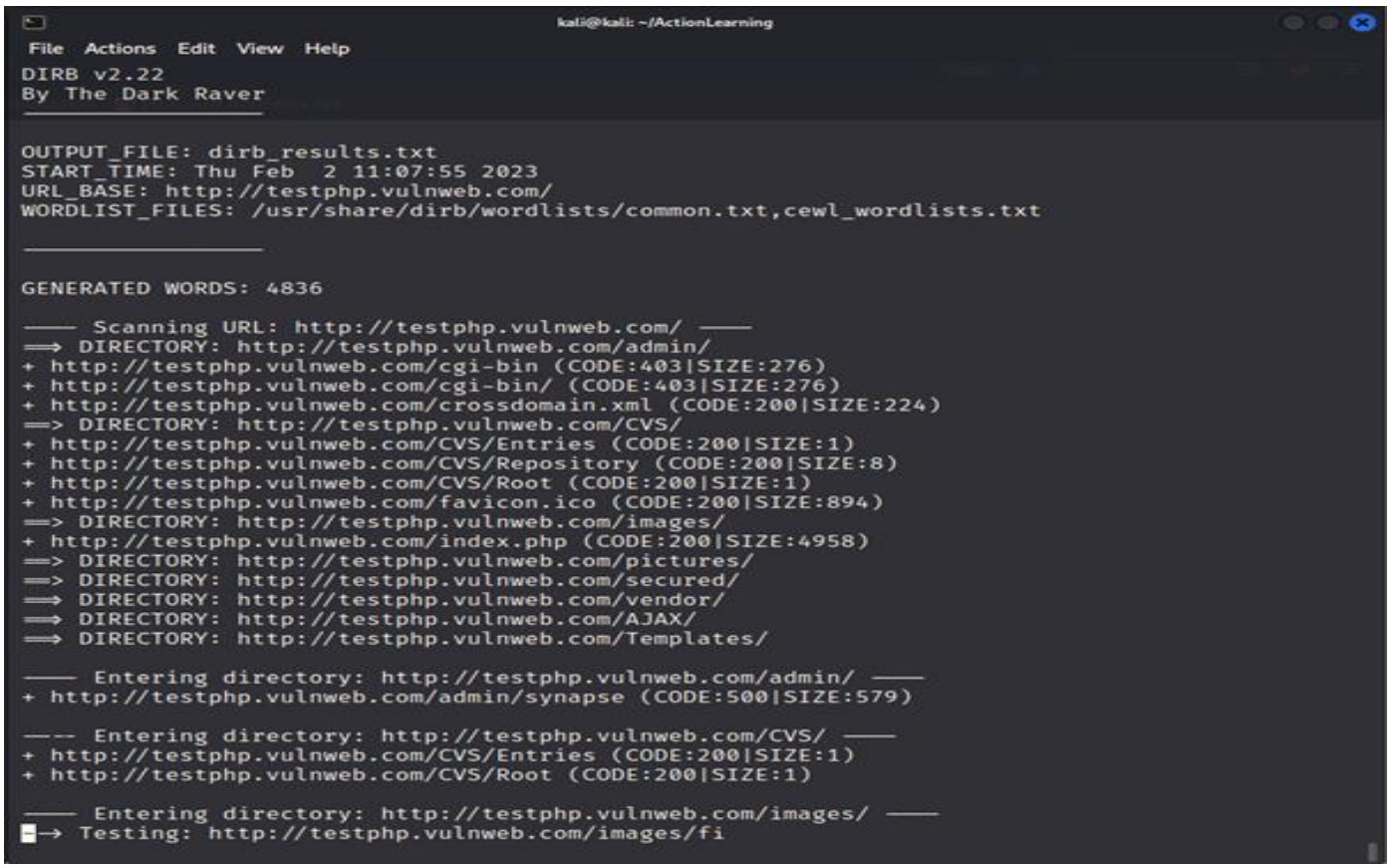


Fig 6: DIRB Executions Result

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24MAY501

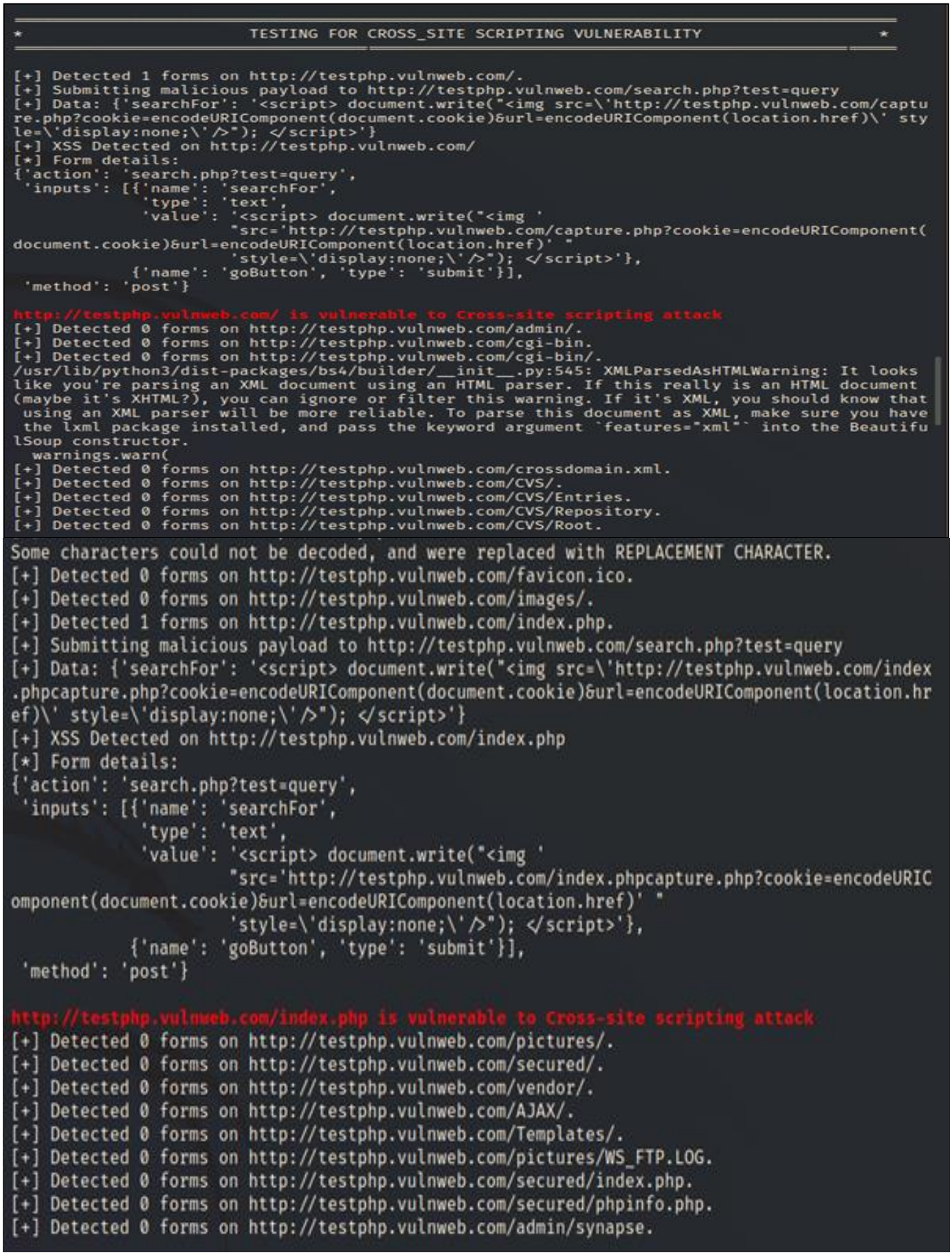


Fig 7: Results of Cross-Site-Scripting

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24MAY501

- *HSTS Header Tester:* Sqlmap also performs tests for more advanced security vulnerabilities. These tests include verifying the availability of the strict-transport-security header, which is an important security feature that helps to protect against man-in-the-middle attacks. It also provides valuable insight into the underlying technologies and components utilized in the creation of the web application, including the version and type of database employed. These tests confirm if the specific security vulnerability is present on the website. Thus, giving the owner an idea of the security of their web application.

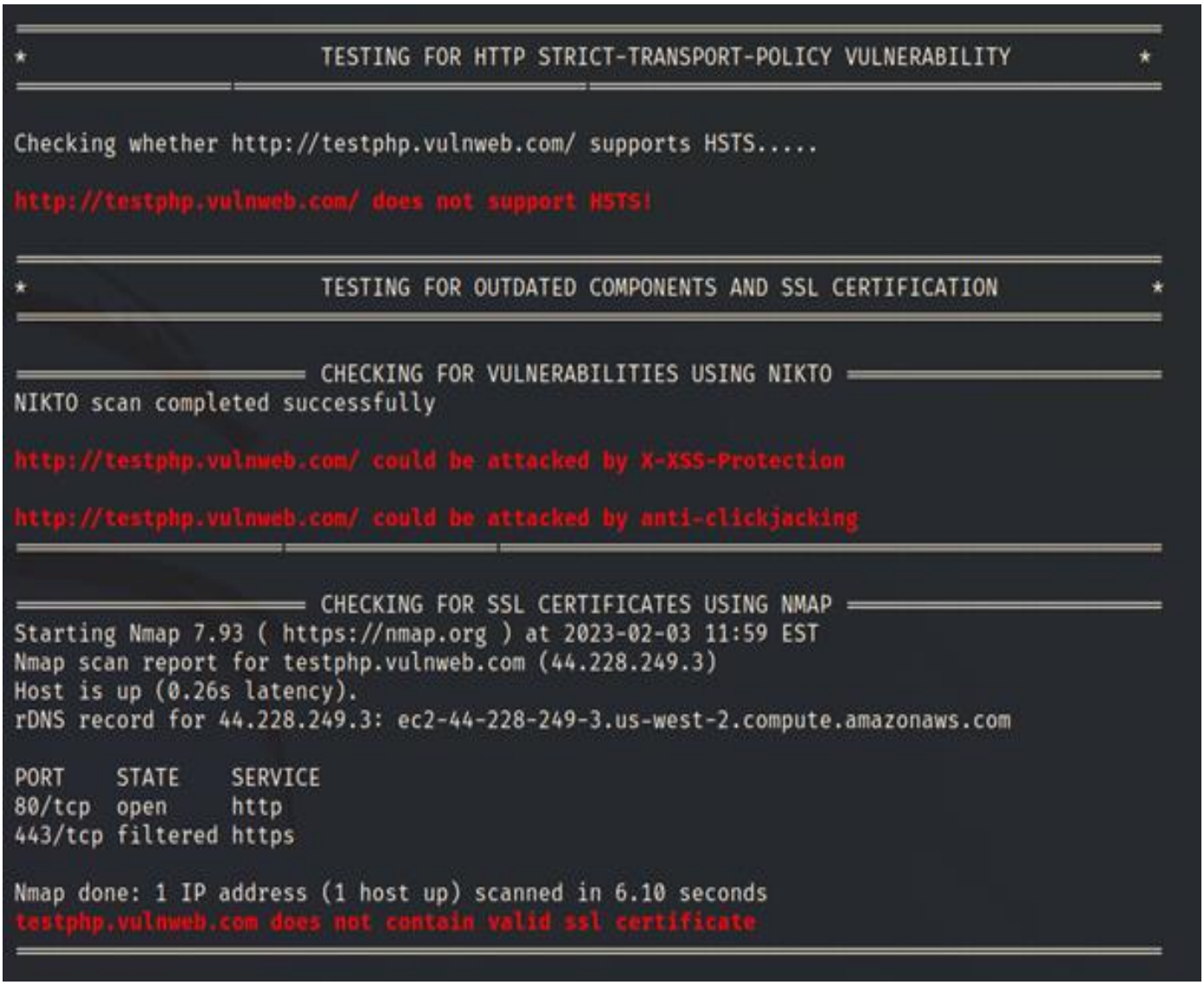
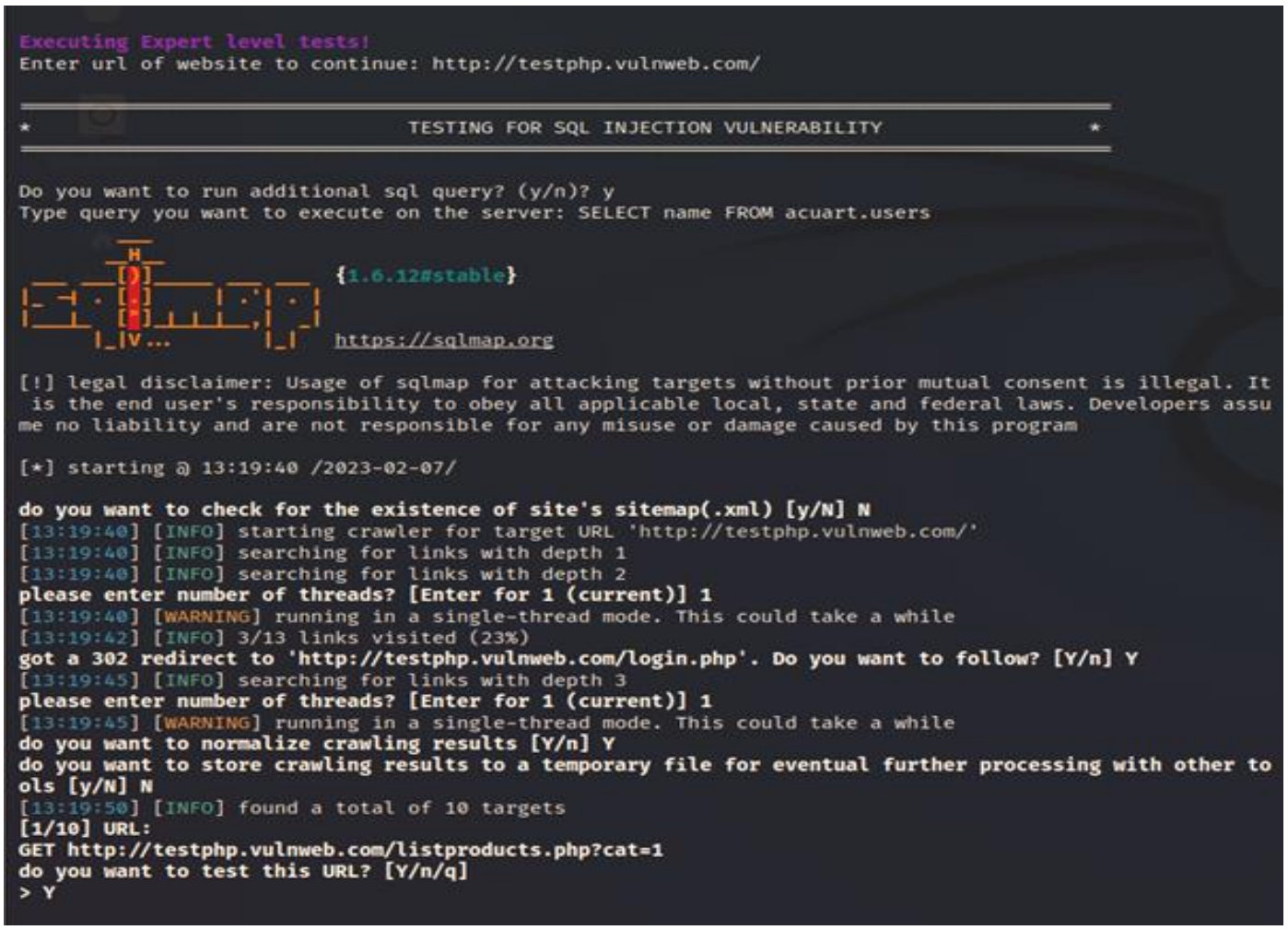
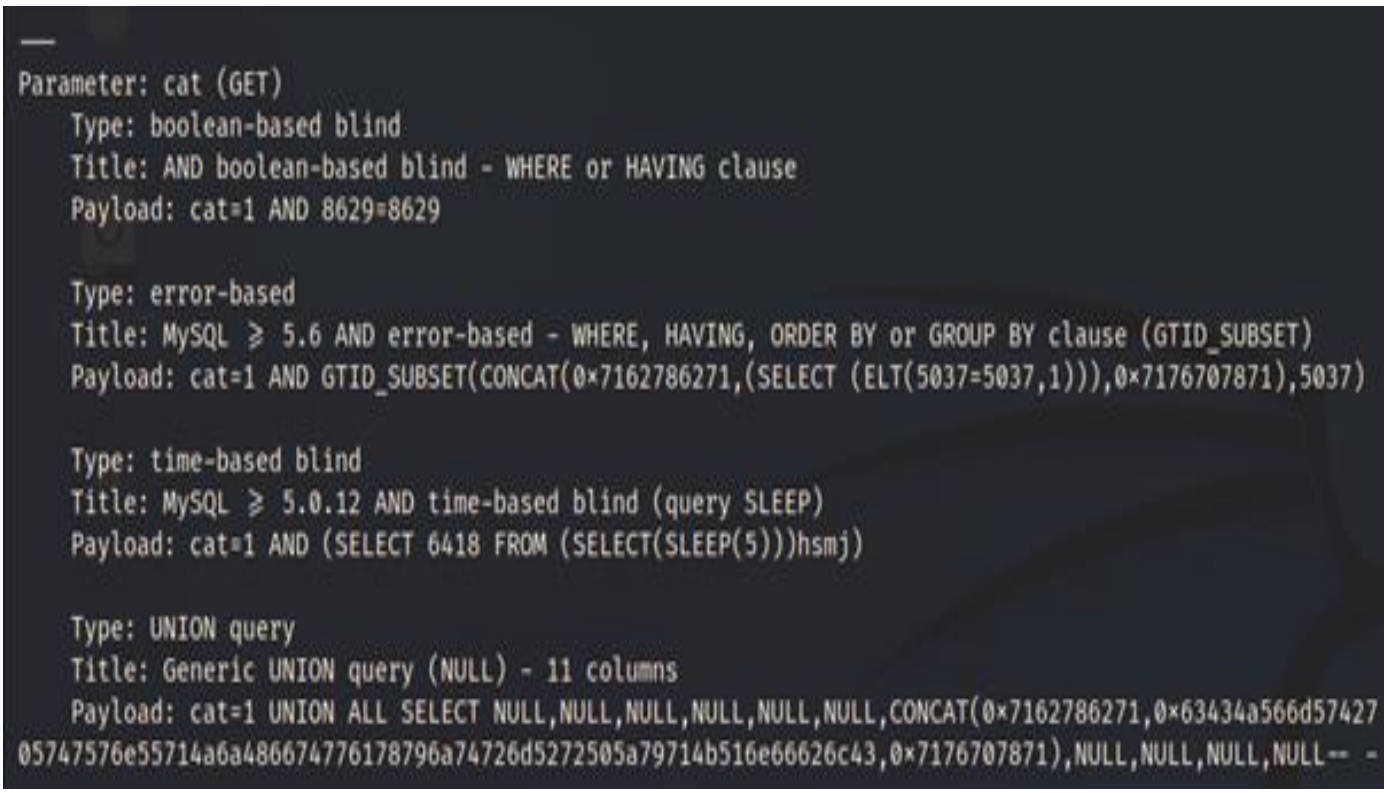


Fig 8: SSL Certification and HSTS Header Test Results for Components

- Valid SSL Certification: Sqlmap also checks for the presence and configuration of SSL certificates, which are used to encrypt communication between the client and server. By verifying the SSL certificate, Sqlmap can identify potential vulnerabilities that could be exploited by an attacker. Other tests that Sqlmap performs include checking for cross-site scripting vulnerabilities, testing for cross-site request forgery, and validating the security of session cookies. Tests for strict-transport-policy header, verification of valid SSL certificate and check for any outdated components are a part of expert level testing too.
- Remaining vulnerabilities testing in Expert level: Remaining tests are like that of the intermediate user level. Cross-site scripting tests are performed for all the hidden files and folders obtained through dirb. Tests for strict-transport-policy header, verification of valid SSL certificate and check for any outdated components are a part of expert level testing too. Expert level users can run any SQL query to test the website using this script. All other functionalities are like intermediary users. User enters a query that he wants to test on the vulnerable server to see the response he receives. Every tool initiated a sub process and the corresponding results were stored in a log file which was further accessed to retrieve information.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24MAY501





ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24MAY501

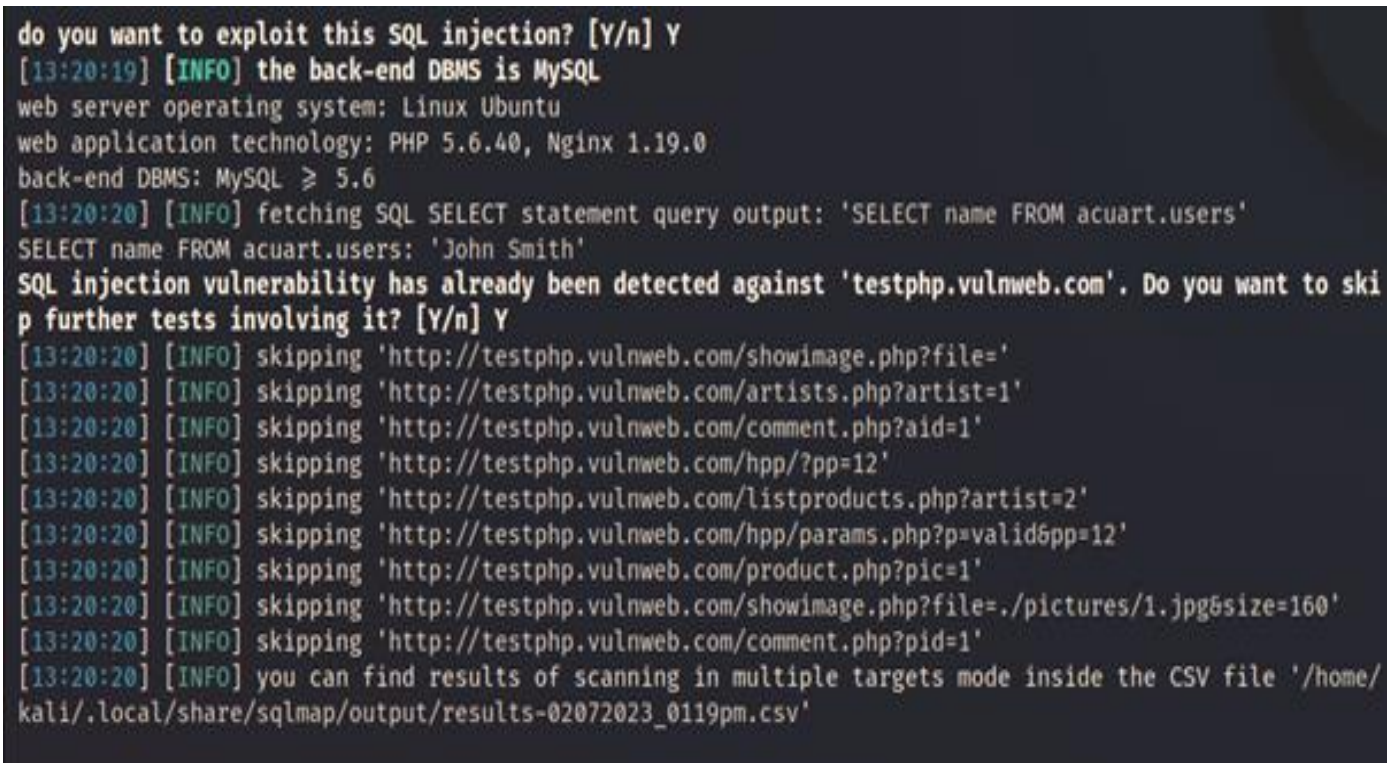


Fig 9: Results of the Remaining Vulnerability in the Expert Level

IV. RESULT

This analysis has provided a thorough examination of the current landscape of SQL injection testing and emphasized the significance of being proactive in protecting against these types of attacks. The increasing prevalence of SQL injection attacks and the potential consequences, such as data breaches, highlight the need for organizations to take these threats seriously. By utilizing the right tools and techniques, as well as implementing secure coding practices, organizations can significantly reduce their risk of falling victim to a SQL injection attack. It is important to note that the threat of SQL injection is constantly evolving, and organizations must stay vigilant and stay up to date with the latest technologies and best practices. SQL injection is a serious security threat that can have devastating consequences for organizations if not properly addressed. The purpose of this thesis was to explore the various methods and techniques used for SQL injection testing along with testing for HTTP Strict Transport Policy header and check for cross-site scripting attacks. In conclusion, by taking a comprehensive and proactive approach to security as it ensures the confidentiality, integrity of their sensitive information.

V. CONCLUSION AND FUTURE WORK

To ensure the web applications' security we were able to test the XSS and HSTS tools that will reduce the chance of website vulnerability by SQL injection attack. For detecting the vulnerabilities this paper discusses the methods and the used tools in testing to identify the loopholes for attacks. Discusses methods and tools for detecting these vulnerabilities. Furthermore, it underscores the importance of incorporating development practices such, as input validation and query usage to thwart SQL attacks. The serious consequences of SQL attacks, including data loss and financial implications underscore why organizations should implement security strategies that involve testing and strong protective measures. The thesis stresses the significance of XSS testing to prevent script injections on web pages and HSTS testing to ensure secure HTTPS transmissions that protect users from attacks and data breaches.

In the future, for easy access, as we want to secure the process of online use as well as computer usage, we are also working to create software for reducing the risk of SQL injection attacks for the webpages. Such a tool would significantly improve an organization's ability to defend against attacks and maintain the confidentiality of their information.

In conclusion, the thesis highlights the importance of taking an approach, to security by staying informed about practices and emerging technologies to address evolving security threats. If this process can be adopted by the people who work in online businesses or use online for their working sites can easily protect their personal information and secure credentials from SQL injection attacks. This can ensure the confidentiality, accessibility, and integrity of the user's information.

ACKNOWLEDGEMENT

We would like to express our gratitude to parents and our friends also dealing with the solution to SQL Injection attack in our research paper, we would thank the group members for contributing their fullest in completing this paper.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24MAY501

REFERENCES

[1]. Alde Alanda, D. S. (September 2021). Web Application Penetration Testing Using SQL Injection. International Journal On Informatics Visualization, 320-326.

[2]. Shobana R, D. M. (2020). A Thorough Study On SQL Injection Attack-Detection And Prevention Techniques And Research Issues. Journal of Information and Computational Science, 135-143.

[3]. Bandi Aruna, B. U. (2020). SQLID Framework In Order To Perceive SQL Injection Attack on Web Application. ICRAEM.

[4]. GitHub. (n.d.). sqlmapproject. Retrieved from GitHub: https://github.com/sqlmapproject/sqlmap

[5]. Invicti. (n.d.). SQL Injection Cheat Sheet. Retrieved from Invicti: https://www.invicti.com/blog/web-security/sql-injection-cheat-sheet/

[6]. Chris Sullo, D. L. (n.d.). Nikto2. Retrieved from CIRT.net: https://cirt.net/Nikto2

[7]. Kali. (n.d.). dirbuster. Retrieved from Kali: https://www.kali.org/tools/dirbuster/

[8]. Malware Bytes. (n.d.). What is SQL Injection. Retrieved from Malware Bytes: https://www.malwarebytes.com/sql-injection

[9]. Wagner, R. (n.d.). How To Test for SQL Injections [Complete Guide]. Retrieved from Code Intelligence: https://www.code-intelligence.com/blog/how-to-test-for-sql-injections

[10]. Moradov, O. (2022, May 12). 5 SQL Injection Test Methods and Why to Automate Your Testing. Retrieved from Bright: https://brightsec.com/blog/sql-injection-test

[11]. nmap.org. (n.d.). Nmap: the Network Mapper. Retrieved from nmap.org: https://nmap.org/

[12]. Abdalla Hadabi, E. E. (March 2022). An Efficient Model to Detect and Prevent SQL Injection Attack. Journal of Karary University for Engineering and Science (JKUES), 141-146.

[13]. OWASP. (n.d.). SQL Injection | OWASP Foundation. Retrieved from OWASP: https://owasp.org/www-community/attacks/SQL_Injection

[14]. OWASP. (n.d.). WSTG - Latest | OWASP Foundation. Retrieved from OWASP: https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/07-Input_Validation_Testing/05-Testing_for_SQL_Injection

[15]. Singh, S. (2022, July 07). Common SQL Injection Attacks. Retrieved from Pentest Tools: https://pentest-tools.com/blog/sql-injection-attacks

[16]. Software Testing Help. (2022, October 25). SQL Injection Testing Tutorial (Example and Prevention of SQL Injection Attack). Retrieved from Software Testing Help: https://www.softwaretestinghelp.com/sql-injection-how-to-test-application-for-sql-injection-attacks/