



XSS/SQL VULNERABILITY SCANNER TOOL (SENTINEL)

Anumandla Dhanush, Ch Arun Teja, Mariserla Manisha, Ruth, G. Aparna

Student/Scholar, Student/Scholar, Student/Scholar, Student/Scholar, Associate Professor
Computer Science and Engineering
(Cybersecurity)

Hyderabad Institute of Technology and Management (HITAM), Gowdavelly(V), Medchal(M), Medchal-Malkajgiri Dist.
501401, Telangana, India

Abstract:

The objective of this paper is to archive Sentinel, a Python-based CLI gadget developed for the automatic identification of Cross-Site Scripting (XSS) and SQL Injection (SQLi) vulnerabilities in the web applications. Proceeding with ParamSpider for parameter detection, Sentinel gets boosted in scanning capacity to provide a much broader spectrum of the existing vulnerabilities. The research starts with a general part on XSS and SQLi vulnerabilities. They are explained why they are significant and how often they affect web applications. Available automated vulnerability scanning tool's unlimitedness is described at first and it is followed by the introduction of Sentinel.

The paper fully explores the built of Sentinel, file's structure, and major features, expressing the purpose of individual parts in the scanning. Instructions for use will be displayed thus consumers can use the product for XSS and SQLi vulnerabilities detection. The results of samples obtained from websites which are at elevated risk have been used as reliable proof that the tool is accurate in identifying and reporting vulnerabilities.

Sentinel's performance is evaluated in comparison to existing scanning tools, and factors like accuracy, clarity, and ease of use are measured. Evaluation results serve as a basis for future improvements and research directions and, therefore, indicate that Sentinel is a smart choice, given its high flexibility as a web security tool.

In the end, Sentinel proves to be a precious ally in the cybersecurity ammunition, providing security professionals and developers with a useful tool which makes it easy to scan and fix the possible non-secure parts of the web applications before the hackers attack the system.

Index Terms:

Cybersecurity, Vulnerability Scanning, Cross-Site Scripting (XSS), SQL Injection (SQLi), Automated Tools, Web Application Security, Command Line Interface (CLI), Python Programming, Parameter Discovery, Threat Detection, Risk Mitigation, Web Security, Exploitation, Vulnerability Management.

1. INTRODUCTION:

As now the internet is the way for web applications to assist people to access and exchange information within the web, ensuring cybersecurity becomes the significant point which remains a matter of high priority in the era of digital. Every time the network, internet access and interconnected networks are the triggers of information leakage, data rearrangement and unauthorized withdrawals from the system. Web app security highly relies on the necessity to guard against Cross-Site Scripting (XSS) and SQL Injection (SQLi) weaknesses, which are the predominant types of security lapses.

XSS are cross-site scripting attacks in which an input is entered into the web page and then runs a malicious script causing a security breach or leaking classified data to another user. SQL Injection (SQLi) rather explores the flaws of a web application database's

security, giving the hackers the privilege to conduct commands through the targets' database and access a host of their sensitive data.

With the purpose of persistently identifying and mitigating these susceptible issues, automated vulnerability scanning tools are the major tool depended on to be effective. To counter the emerging threat environment, this paper suggests a Sentinel CLI tool built using Python for the automated detection of both XSS and SQLi vulnerabilities in web apps.

Sentinel is now reliant on paramspider, a paraphrase learning parametrize algorithm, for increased scanning functionality, covering a broader spectrum of possible vulnerabilities. Privacy professionals and developers are granted a unique and user-friendly tool, which consequently leaves businesses no choice but to take control of the risks posed by web application bugs.

This piece seeks to cover Sentinel's institutionalization into on-campus activities comprehensively, both in terms of process and outcome. This starting section is focused on XSS and SQLi vulnerabilities with detailed information about their implications for web programs of current engineering alongside frequency. The research, after that, surveys the vulnerability scanning tools in existence, so that Sentinel is introduced in the midst of the given context. The details on Sentinel's implementation, application and testing procedures are presented, which clarify the usability and efficiency of Sentinel in the accurate detection and reporting of vulnerabilities.

2. LITERATURE SURVEY:

Thus, the primary focus of the study is on the application of web security research developments in cybersecurity literature to provide novel solutions for XSS and SQLi which are still very dangerous and could be disastrous. Although papers in the Safety and Security section of the American Industrial Hygiene Association journal, namely Klein et al. (2007) and Halfond et al. (2006), suggest that XSS is ubiquitous and one of the most common vulnerabilities, yet it goes frequently undetected.

Autonomous scanner tools for vulnerabilities become a game changer as they are the most valuable weapon behind the web application vulnerability war. Hence, the relevancy of these tools for the disclosure of exploits such as XSS, also known as Huang et al. and Huang and Liu, have been proved to be effective in boundary detection. Since textbooks are now digital, other progress has been initiated. It is also true that there were plenty of studies led by Klein et al. (2007) and Huang et al. (2010) that were about the idea of the establishment of unique strategies for the scanning of vulnerabilities, resulting in the production of best practices and the new ways to optimize the performance.

Besides, the research by Loper and others (2015) was undertaken to assess the capability of the app development to track down web parameters based on Sentinel, the web app security vulnerability scanner; in addition, ParamSpider was included as a researcher. The single approach for ParamSpider integration into vulnerability scanning tools is another thing the scientists dealing with security discuss. They do that and say that using this approach the process of scanning can be completed in a faster and better way.

3. PURPOSE AND SCOPE:

The main task for the vulnerability scanning tool is to conduct automated detection of Cross-Site Scripting (XSS) and SQL Injection (SQLi) vulnerabilities in web applications. It aspires to make web application security better by providing a powerful but easy to use tool for finding security holes. The tool's functionality comprises comprehensive scanning features, convenient integration and use, and the ability to be customized to suit individual security requirements.

- **Automated Vulnerability Detection:** The goal of the project is to build a CLI-based tool that can be used to automatically detect Cross-Site Scripting (XSS) and SQL Injection (SQLi) flaws in web applications. By using the automated scanning techniques, the tool seeks to shorten the identification process of security gaps, therefore improving web applications security status.
- **Comprehensive Scanning Capabilities:** The project scope includes the creation of a thorough scanning tool that has the capability to detect both XSS and SQLi vulnerabilities present across diverse web applications. The tool should be designed to have deep and broad parameter discovery with payload injection to guarantee the least number of false negatives.
- **Ease of Integration and Use:** The tool should be developed to be user-friendly and easy to integrate, and so that even security professionals and developers with different technical backgrounds can use it. Its CLI must have clear-cut commands and options for initiating scans, viewing results, and editing scanning parameters.
- **Flexibility and Customization:** The main objective of the project is to provide flexibility and adaptability to the users by letting them configure the scanning parameters according to their needs and demands. On the one hand, the program can perform single scans for XSS or SQLi vulnerabilities separately and on the other hand can adjust scanning parameters and payloads depending on the situation.

Ease of Use:

- Sentinel provides the users with a user-friendly CLI interface which is easily understandable by the users with various levels of technical background.
- The software offers simple and straightforward instructions for performing scans, displaying the results, and adjusting the parameters of the scanning process.
- The step-by-step installation process and methods of effective scanning are explained in detail with user guide instructions.

4. METHODOLOGY:

- **Requirement Analysis:** Lay out the necessary features for the vulnerability scanning tool, such as the identification of XSS and SQLi vulnerabilities, parameter discovery, and user-friendliness.
- **Design Architecture:** Create the structure of the tool together with the elements like the command-line interface (CLI), modules for XSS and SQLi scanning, and integration with ParamSpider for parameter finding.
- **Implementation:** Develop the tool in Python, using libraries like requests, colorama, and paramspider, and follow the rules and standards in coding and documentation.
- **Testing:** Execute comprehensive testing to ensure the tool is capable of detecting XSS and SQLi vulnerabilities across different web apps.
- **Integration:** Parameter Discovery Tool ParamSpider is integrated to facilitate the smooth functioning of the tool and to increase the scanning capability.
- **User Feedback and Iteration:** Get users and key players feedback, make the tool design and functionality changes in accordance with user experience and needs.
- **Documentation and Deployment:** Present comprehensive documentation, such as an installation manual, a user guide, or technical specifications, and release the tool for public use, making it available and easy to use.

5. REQUIREMENTS AND INSTALLATION:**a) Software Requirements:**

- **Python Interpreter:** Python interpreter (Python 3. x) is necessary to run the code of the project.
- **ParamSpider:** The project uses ParamSpider, an in-built parameter identification tool, for finding parameters in web applications.
- **Dependencies:** Install the required dependencies as listed in the requirements. txt file through pip, which is Python's package manager.
- **Operating System:** The project is arranged in a way that it is compatible with different operating systems such as Windows, macOS and Linux distributions.

b) Hardware Requirements:

- **Processor:** Handle scanning tasks efficiently using a multi-core processor with sufficient power of processing.
- **Memory (RAM):** At least 4GB of RAM is the minimum requirement to ensure the smooth running of the vulnerability scanning tool.
- **Storage:** Sufficient disk quota to keep the project files, dependencies, and any scan outcomes produced during the process
- **Internet Connection:** The internet connection, which is active, is needed to fetch dependencies, use web applications for scanning, and obtain updates or additional resources during the project's execution.

c) Operating System Requirements:

- **Windows:** The project is compatible with most Windows operating systems versions such as Windows 7, 8, 10 and 11.
- **macOS:** The application is compatible with macOS High Sierra, Mojave, and Catalina.
- **Linux:** Our project is compatible with different Linux distributions, including Ubuntu, Fedora, and CentOS.

The compatibility of the project with multiple operating systems is the one factor that makes the vulnerability scanning tool easy to use on different platforms.

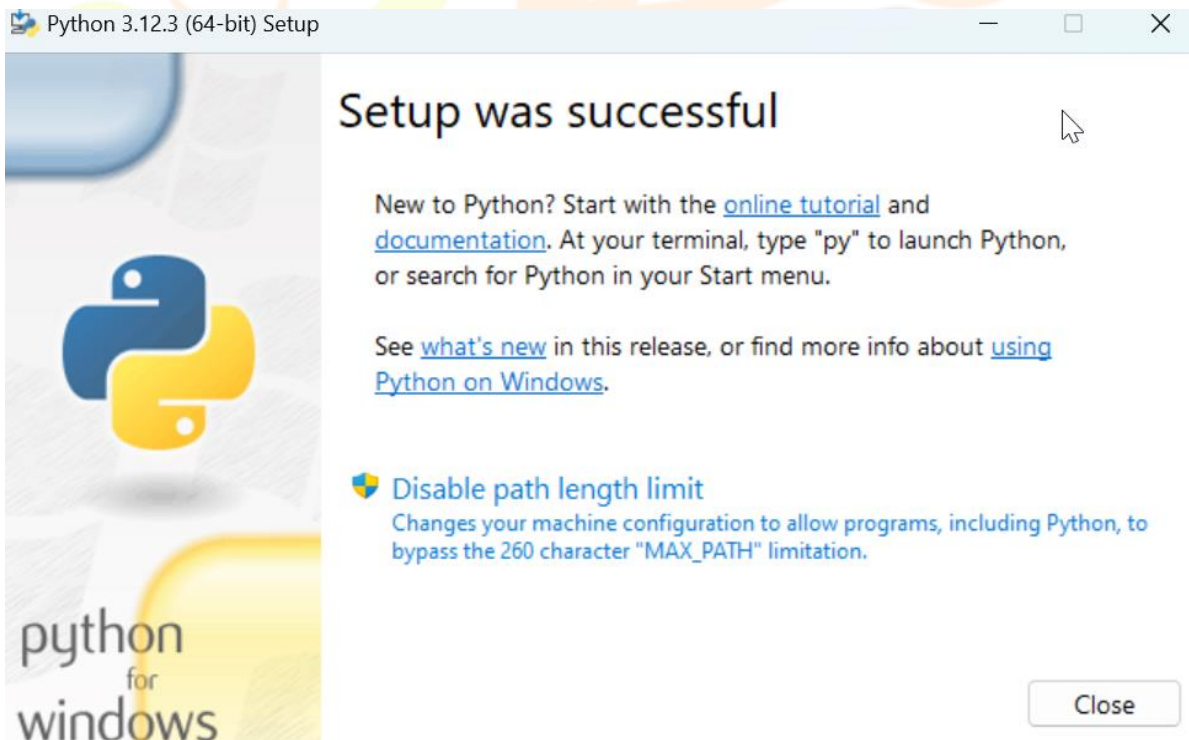
d) Installation:

Download and set up the latest version of Python 3.x from the official website of python(<https://www.python.org/>).

Step 1: Get the python installer from its official website.

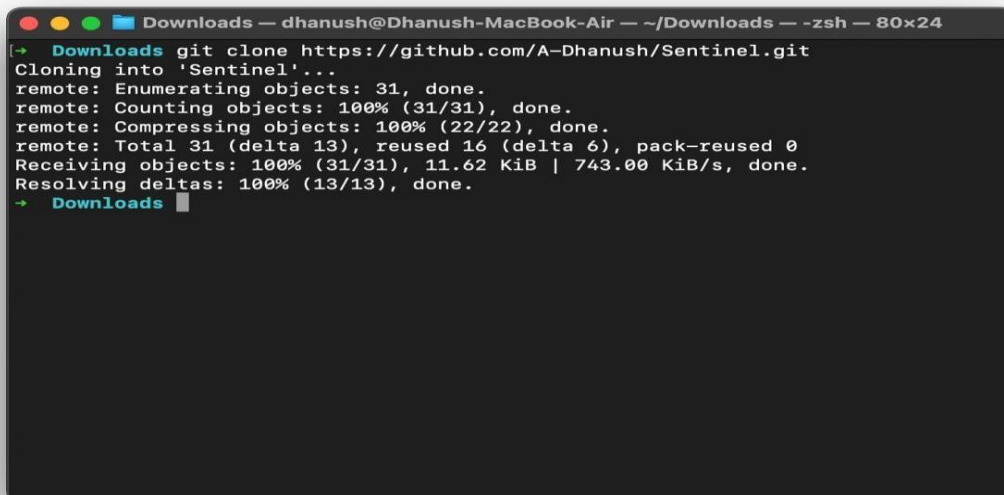


Step 2: After installing of python setup then,



Step 3: Clone the project repository from the version control system by using the git clone command or download the project as a zip file and extract it onto a directory of your choice.

Step 4: Open a terminal or command prompt and navigate to the directory where you cloned or extracted the project.



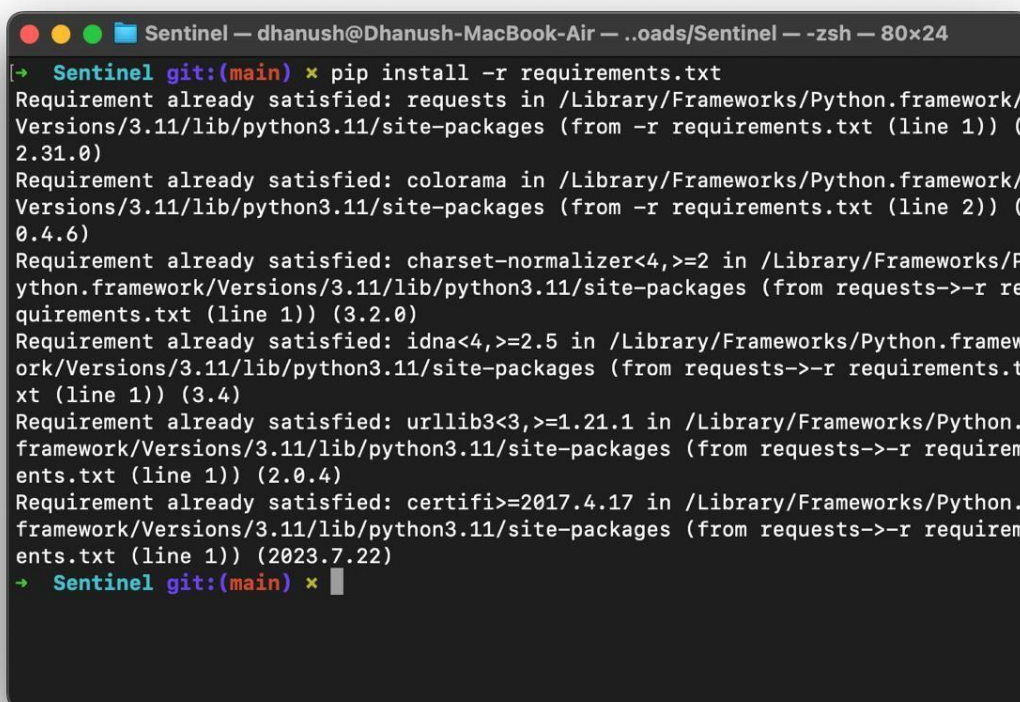
```

Downloads — dhanush@Dhanush-MacBook-Air — ~/Downloads — zsh — 80x24
[→ Downloads git clone https://github.com/A-Dhanush/Sentinel.git
Cloning into 'Sentinel'...
remote: Enumerating objects: 31, done.
remote: Counting objects: 100% (31/31), done.
remote: Compressing objects: 100% (22/22), done.
remote: Total 31 (delta 13), reused 16 (delta 6), pack-reused 0
Receiving objects: 100% (31/31), 11.62 KiB | 743.00 KiB/s, done.
Resolving deltas: 100% (13/13), done.
→ Downloads █

```

Step 5: Run the following command to install the required dependencies listed in the requirements.txt file.

Command prompt code- 'pip install -r requirements.txt'



```

Sentinel — dhanush@Dhanush-MacBook-Air — ..oads/Sentinel — zsh — 80x24
[→ Sentinel git:(main) * pip install -r requirements.txt
Requirement already satisfied: requests in /Library/Frameworks/Python.framework/
Versions/3.11/lib/python3.11/site-packages (from -r requirements.txt (line 1)) (
2.31.0)
Requirement already satisfied: colorama in /Library/Frameworks/Python.framework/
Versions/3.11/lib/python3.11/site-packages (from -r requirements.txt (line 2)) (
0.4.6)
Requirement already satisfied: charset-normalizer<4,>=2 in /Library/Frameworks/P
ython.framework/Versions/3.11/lib/python3.11/site-packages (from requests->-r re
quirements.txt (line 1)) (3.2.0)
Requirement already satisfied: idna<4,>=2.5 in /Library/Frameworks/Python.frame
ork/Versions/3.11/lib/python3.11/site-packages (from requests->-r requirements.t
xt (line 1)) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /Library/Frameworks/Python.
framework/Versions/3.11/lib/python3.11/site-packages (from requests->-r requirem
ents.txt (line 1)) (2.0.4)
Requirement already satisfied: certifi>=2017.4.17 in /Library/Frameworks/Python.
framework/Versions/3.11/lib/python3.11/site-packages (from requests->-r requirem
ents.txt (line 1)) (2023.7.22)
→ Sentinel git:(main) * █

```

Step 6: Execute the sentinel.py file with Python and provide the target URL as a command-line argument.

6. MODEL AND ARCHITECTURE:

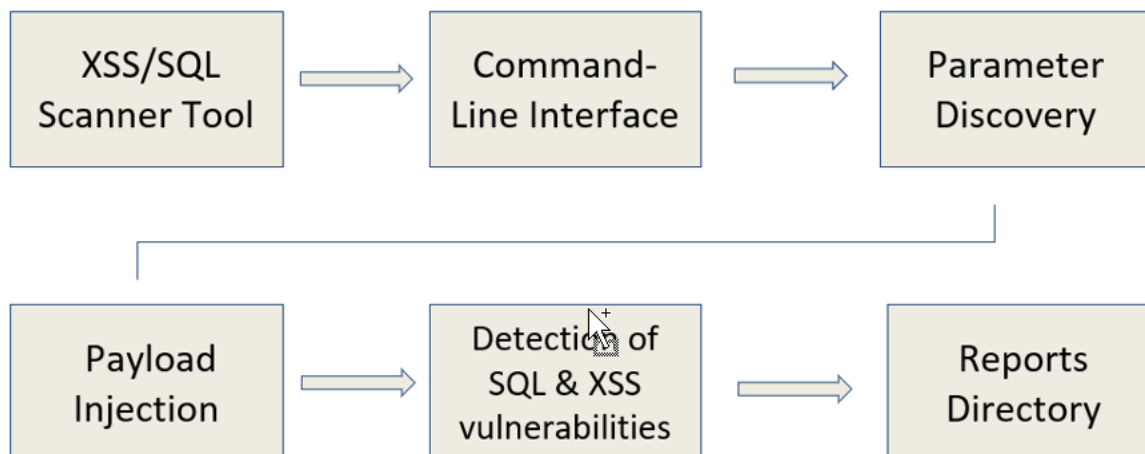


FIG: BLOCK DIAGRAM

- **Input Module:** Takes the command line input from the user, validates, and parses it for use in the scanning process.
- **Scanner Module:** Performs XSS and SQL injections scans on targeted web applications by using parameter discovery, payload injection and response analysis techniques.
- **Vulnerability Classification Module:** Analyzes the scan results to assign the type of identified vulnerabilities (XSS or SQLi) and the severity by using set rules or heuristics.
- **Report Generation Module:** Write complete Findings Reports, detailing scan results, such as vulnerability classifications, location, severity and more for easy remediation.

7. MODEL AND ARCHITECTURE:

Execute the sentinel.py file with Python and provide the target URL as a command-line argument.

```

1 import subprocess
2 import requests
3 import argparse
4 import re
5 from xss import start_xss_scan
6 from sqli import start_sqli_scan
7 from concurrent.futures import ThreadPoolExecutor
8 from threading import Lock
9 from colorama import Fore
10
11 print(Fore.LIGHTBLUE_EX + """
12
13 ██████████
14 ██████████
15 ██████████
16 ██████████
17 ██████████
18
19 #XSS/SQL Vulnerability Scanner
20 """ + Fore.WHITE)
21
22 if __name__ == "__main__":
23     parser = argparse.ArgumentParser(description="XSS/SQL Vulnerability Scanner")
24     parser.add_argument("-u", "--url", required=True, help="Target URL")
25     parser.add_argument("--xss", action="store_true", help="Scan only for XSS vulnerabilities")
26     parser.add_argument("--sql", action="store_true", help="Scan only for SQL vulnerabilities")
27     args = parser.parse_args()
28
29     if args.xss:
30         start_xss_scan(args.url)
  
```

```

Sentinel — dhanush@Dhanush-MacBook-Air — ..ject/Sentinel — -zsh — 177x34
+ Sentinel git:(main) x python3 sentinel.py -u testphp.vulnweb.com

#XSS/SQL Vulnerability Scanner

[+] Testing http://testphp.vulnweb.com/hpp/params.php?aaaa%2F=<script>alert('XSS')</script>&p=<script>alert('XSS')</script>&pp=<script>alert('XSS')</script>
[+] Testing http://testphp.vulnweb.com/listproducts.php?artist=<script>alert('XSS')</script>&mp%3Basdf=<script>alert('XSS')</script>&mp%3Bcat=<script>alert('XSS')</script>
[+] Testing http://testphp.vulnweb.com/Mod_Rewrite_Shop/details.php?id=<script>alert('XSS')</script>&mjqN=<script>alert('XSS')</script>
[+] Testing http://testphp.vulnweb.com/product.php?pic=<script>alert('XSS')</script>&hNi=<script>alert('XSS')</script>
[+] Testing http://testphp.vulnweb.com/search.php?test=<script>alert('XSS')</script>&cat=<script>alert('XSS')</script>&ttl=<script>alert('XSS')</script>
[+] Testing http://testphp.vulnweb.com/showimage.php?file=<script>alert('XSS')</script>
[+] Testing http://testphp.vulnweb.com/AJAX/infoartist.php?id=<script>alert('XSS')</script>&fYij=<script>alert('XSS')</script>
[+] Testing http://testphp.vulnweb.com/redis.php?=<script>alert('XSS')</script>&user=<script>alert('XSS')</script>
[+] Testing http://testphp.vulnweb.com/artists.php?artist=<script>alert('XSS')</script>
[+] Testing http://testphp.vulnweb.com/trk=<script>alert('XSS')</script>
[+] Vulnerable: http://testphp.vulnweb.com/hpp/params.php?aaaa%2F=<script>alert('XSS')</script>&p=<script>alert('XSS')</script>&pp=<script>alert('XSS')</script>
[+] Vulnerable: http://testphp.vulnweb.com/listproducts.php?artist=<script>alert('XSS')</script>&mp%3Basdf=<script>alert('XSS')</script>&mp%3Bcat=<script>alert('XSS')</script>
[+] Testing http://testphp.vulnweb.com/artist.php?artist=<script>alert('XSS')</script>
[+] Testing http://testphp.vulnweb.com/admin/?C=<script>alert('XSS')</script>
[+] Testing http://testphp.vulnweb.com/login.php?id=<script>alert('XSS')</script>&ntFH=<script>alert('XSS')</script>
[+] Testing http://testphp.vulnweb.com/Mod_Rewrite_Shop/rate.php?id=<script>alert('XSS')</script>
[+] Testing http://testphp.vulnweb.com/index.php?%25id%25=<script>alert('XSS')</script>&user=<script>alert('XSS')</script>
[+] Testing http://testphp.vulnweb.com/secured/phpinfo.php?=<script>alert('XSS')</script>
[+] Vulnerable: http://testphp.vulnweb.com/showimage.php?file=<script>alert('XSS')</script>
[+] Testing http://testphp.vulnweb.com/hpp/index.php?pp=<script>alert('XSS')</script>
[+] Testing http://testphp.vulnweb.com/comment.php?aid=<script>alert('XSS')</script>
[+] Testing http://testphp.vulnweb.com/bxss/vuln.php?id=<script>alert('XSS')</script>
[+] Testing http://testphp.vulnweb.com/Mod_Rewrite_Shop/buy.php?id=<script>alert('XSS')</script>
[+] Vulnerable: http://testphp.vulnweb.com/hpp/index.php?pp=<script>alert('XSS')</script>

```

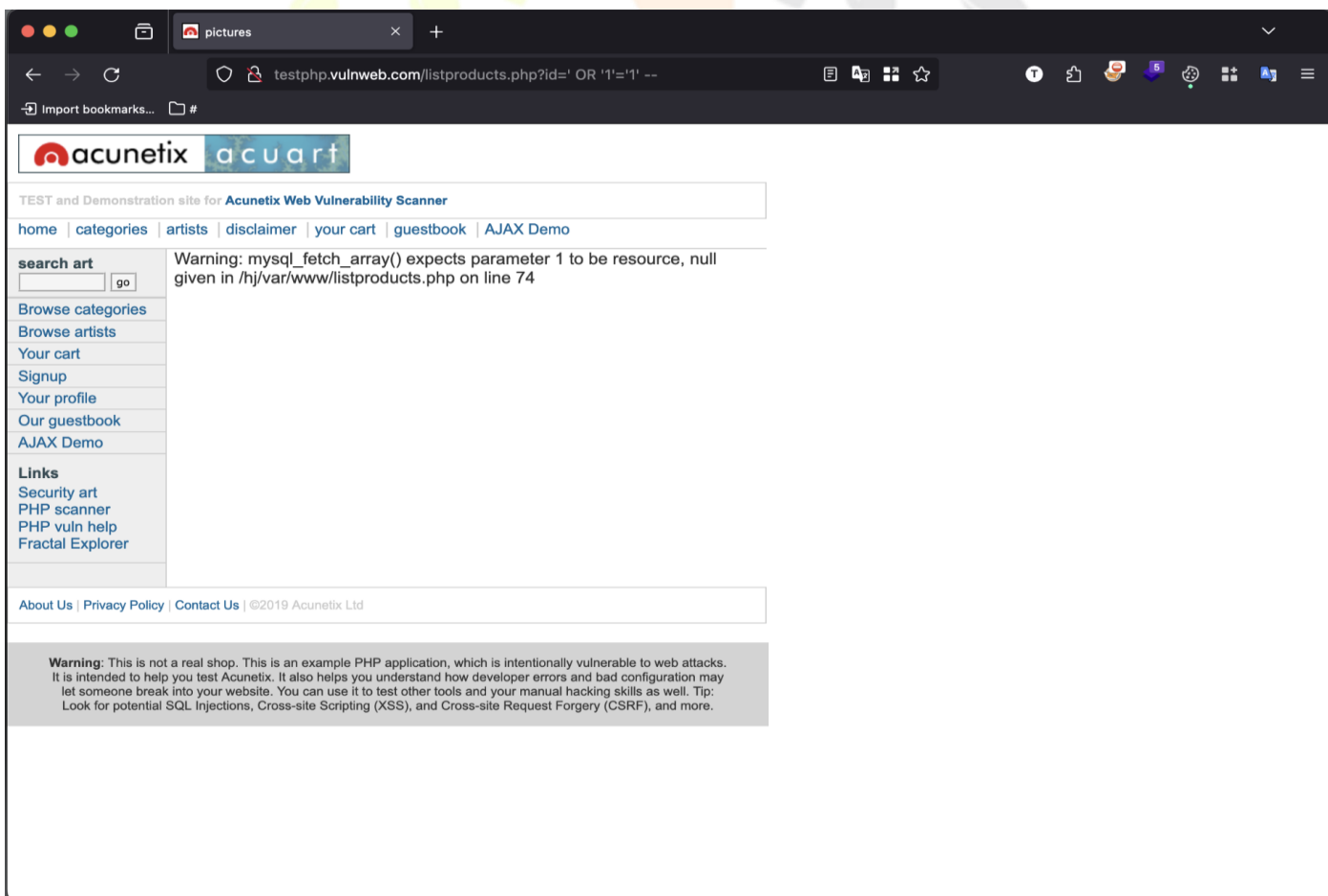
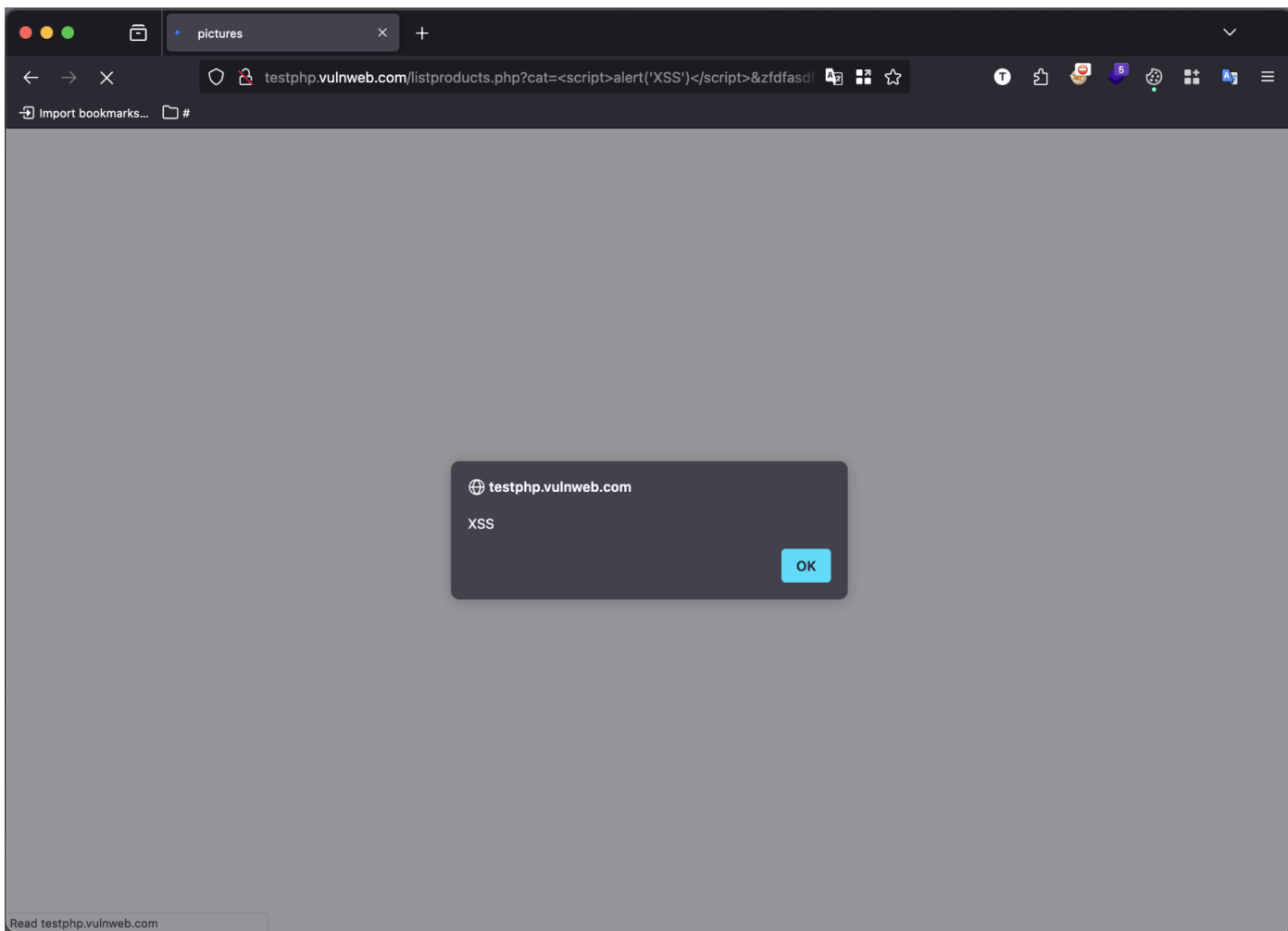
After the scan is completed, review the results in the report.txt file generated by the tool. This file will contain information about any vulnerabilities detected during the scan.

```

report.txt
XSS Vulnerable: http://testphp.vulnweb.com/listproducts.php?cat=<script>alert('XSS')</script>&zfdfasdf=<script>alert('XSS')</script>
XSS Vulnerable: http://testphp.vulnweb.com/hpp/?pp=<script>alert('XSS')</script>
XSS Vulnerable: http://testphp.vulnweb.com/hpp/index.php?pp=<script>alert('XSS')</script>
XSS Vulnerable: http://testphp.vulnweb.com/showimage.php?file=<script>alert('XSS')</script>&size=<script>alert('XSS')</script>
DBMS Detected: MySQL
SQL Vulnerable: http://testphp.vulnweb.com/AJAX/infoartist.php?id=' OR '1'='1' --&YVeN=' OR '1'='1' --
SQL Vulnerable: http://testphp.vulnweb.com/artists.php?+artist=' OR '1'='1' --
SQL Vulnerable: http://testphp.vulnweb.com/product.php?pic=' OR '1'='1' --
SQL Vulnerable: http://testphp.vulnweb.com/AJAX/infoartist.php?id=' OR '1'='1' --&fYij=' OR '1'='1' --
SQL Vulnerable: http://testphp.vulnweb.com/AJAX/infocateg.php?id=' OR '1'='1' --
SQL Vulnerable: http://testphp.vulnweb.com/listproducts.php?cat=' OR '1'='1' --&zfdfasdf=' OR '1'='1' --
SQL Vulnerable: http://testphp.vulnweb.com/bxss/vuln.php?id=' OR '1'='1' --
SQL Vulnerable: http://testphp.vulnweb.com/listproducts.php?id=' OR '1'='1' --
SQL Vulnerable: http://testphp.vulnweb.com/search.php?test=' OR '1'='1' --&cat=' OR '1'='1' --&ttl=' OR '1'='1' --
SQL Vulnerable: http://testphp.vulnweb.com/artists.php?artist=' OR '1'='1' --
SQL Vulnerable: http://testphp.vulnweb.com/Mod_Rewrite_Shop/details.php?id=' OR '1'='1' --
SQL Vulnerable: http://testphp.vulnweb.com/Mod_Rewrite_Shop/buy.php?id=' OR '1'='1' --
SQL Vulnerable: http://testphp.vulnweb.com/search.php?test=' OR '1'='1' --
SQL Vulnerable: http://testphp.vulnweb.com/AJAX/infoartist.php?id=' OR '1'='1' --
SQL Vulnerable: http://testphp.vulnweb.com/listproducts.php?artist=' OR '1'='1' --&mp%3Basdf=' OR '1'='1' --&mp%3Bcat=' OR '1'='1' --
SQL Vulnerable: http://testphp.vulnweb.com/listproducts.php?artist=' OR '1'='1' --&sd=' OR '1'='1' --&cat=' OR '1'='1' --
SQL Vulnerable: http://testphp.vulnweb.com/AJAX/infoartist.php?id=' OR '1'='1' --&DUMM=' OR '1'='1' --
SQL Vulnerable: http://testphp.vulnweb.com/Mod_Rewrite_Shop/rate.php?id=' OR '1'='1' --
SQL Vulnerable: http://testphp.vulnweb.com/listproducts.php?artist=' OR '1'='1' --
SQL Vulnerable: http://testphp.vulnweb.com/listproducts.php?cat=' OR '1'='1' --
SQL Vulnerable: http://testphp.vulnweb.com/product.php?pic=' OR '1'='1' --&hNi=' OR '1'='1' --

```

When the report file is clicked on it will be shown in the web browser and the information it contains would be the target URL and the date of scan and as well as the risk classifications and the detected SQL and XSS vulnerabilities.



XSS (Cross-Site Scripting) and SQL vulnerabilities are detected through the vulnerability scanner tool.

CONCLUSION:

As a tremendous addition to web application security, the vulnerability scanner tool development has been successful. Using the latest methodologies of scanning and comprehensive reporting offered by the tool, security professionals and developers are able to detect and take appropriate actions to eliminate XSS and SQLi vulnerabilities efficiently. The scanning process is automated and vulnerability classification is simplified; therefore, the tool is a good way to strengthen the security position of web applications.

The factor that defines the project is the combination and integration of advanced algorithms with easy-to-use features like a command-line interface and customizable reports, which make it available to a large section of users. Besides that, the integration of ParamSpider for parameter discovery improves the comprehensiveness and completeness of the web application scanning.

The project has laid the foundation for future expansion in terms of vulnerability detection methods, incorporation of more frameworks, and enhanced reporting capabilities. Moreover, periodic maintenance and upgrades will be necessary to address the new security threats and keep the tool topical for an ever-changing cybersecurity field.

To sum up, vulnerability scanner tool project being evidence that proactive security measures are indispensable in protecting web applications against the bad guys. The main aim of this tool is to equip people with the ability to detect and resolve weaknesses beforehand. The goal is to build a safer and more stable digital environment.

ACKNOWLEDGMENTS:

We would like to thank our guide Aparna ma'am for her important suggestions to improve the standard of the paper. We are also grateful to her for helping us review our performance regularly. We would also like to thank the Department of Computer Science Engineering (Cyber Security), HITAM, Hyderabad.

REFERENCES:

- Jim Manico, August Detlefsen, and Steve Jaworski. "OWASP Top Ten". Open Web Application Security Project (OWASP). Available online: <https://owasp.org/www-project-top-ten/>
- Daniel Miessler. "Cross-Site Scripting (XSS)". Available online: <https://owasp.org/www-community/attacks/XSS/>
- PortSwigger. "SQL Injection". Available online: <https://portswigger.net/web-security/sql-injection>

