

WEB APPLICATION VULNERABILITY SCANNER

ABSTARCT:

In this project of web application vulnerability scanner, a scanner is designed such that it can scan for the vulnerabilities in a website. This scanner is designed using python language. This scanner focuses on three main vulnerability issues that are a common path for the attackers to get into a website. The three main issues that interest this scanner are: cross site scripting (XSS), SQL injection (SQLi), and missing CSRF protection. This scanner crawls into the target website, it identifies the inputs and injects the crafted payloads into it. The scanner, then observes the response of the target website, to the injected payloads and analyses server responses for error patterns. It demonstrates an automatic security testing. It uses open end libraries like *requests* and *BeautifulSoup*, that can be used to further develop this project into a proper professional pen-testing (penetration testing) framework.

INTRODUCTION:

The web applications are the main components of present time digital infrastructure. These web applications are very useful in providing an accessible, reliable, and effective services among and across the devices. These services include, e-commerce, online banking, online searches, real-time data analysis etc. These web applications, come along with many vulnerabilities even after being an important part of the digital framework. These vulnerabilities are the loopholes that gives unauthorized access to the attackers to barge into the web application in order to perform illegal tasks for their benefit (like stealing victim's personal information to use it against the victim etc.). These vulnerabilities are basically the open ports without proper security. The main vulnerabilities that can be verified and detected using this scanner are- cross site scripting (XSS), SQL injection (SQLi), and missing CSRF protection. XSS allows the attacker to inject malicious code or scripts into trusted websites, helping him/her to steal the data by running in the background of the browser. SQLi allows the attacker to interfere with the application queries and gets access to the main database. CSRF protection is security measure that XSS attacks by ensuring that state changing requests are originating from a trustable source. These vulnerabilities allow the attacker to steal data, modify the application appearance and bypass the authentication. Detecting these vulnerabilities manually requires more expertise and time, thus, simple automated vulnerability scanners are designed generally. In this project, a similar web application scanner is built using python language. This scanner crawls into a website, sends payloads, and collects internal links. Then it scans and analyses the website's response to those payloads for any kind of errors. Thus, the vulnerabilities like; XSS, SQLi and CSRF protection failures can be detected.

TOOLS:

Python is used for the purpose of programming. HTML parsing library used is 'BeautifulSoup', HTTP library used is 'requests', standard library modules used are 'urllib.parse'. These are common are not too high-end tools used to create a basic version of automated web application vulnerability scanner.

STEPS USED TO BUILD THIS SCANNER:

This scanner was built by careful execution of the code and usage of various libraries. The 'requests' HTTP library was used to GET and POST requests or payloads to the target website. The 'BeautifulSoup' HTML parsing library was used to parse through the html links of the website to understand inputs and analyse the forms. The 'urllib.parse' standard library module is used for the query manipulation in order to detect any error in the messages. These tools helped in simple expression of this python-based web application vulnerability scanner. The Steps used to build this scanner include:

1. Requirement Analysis and Design: Firstly, the scanner's focus was defined; meaning that this scanner will be focussing on detecting any the three following vulnerabilities - reflected XSS, basic SQL Injection indicators, and missing CSRF tokens. A simple framework of the project is built. The framework consists of a crawler function that visits the internal links of the target website, a form extraction function that collects forms and their input fields, a testing function for detecting XSS, SQLi and CSRF security compromise.
2. Implementing the Web Crawler: The crawler utilizes the 'requests' library to download every page and the 'BeautifulSoup' is used to parse those downloaded pages. All the <a> tags with href attributes are collected. The anchor or <a> tags with its href attributes are used to create hyperlink to web pages, email addresses etc, present on the same page. The crawler maintains a set of URLs that are already visited at that time, thus avoiding the repeated scanning or the loop scanning of the same sites. A depth limit is also used to restrict the depth of the scan.
3. Extracting Forms and Inputs: For all the pages visited, the program hunts for <form> elements. For each form, action URL, HTTP (GET/POST) method and the input field names are recorded. This information is used by the programme later to send test requests with crafted payloads.
4. Implementing the XSS Scanner: To test for Cross-Site Scripting, a simple JavaScript payload such as <script>alert(1)</script> is injected into each input field of the form. The response page is then explored to observe if the payload appeared directly in the returned HTML. If it is reflected unencoded, the form is flagged as potentially vulnerable to reflected XSS.
5. Implementing the SQL Injection Scanner: The SQL Injection tests use general payloads like ' OR '1'='1 or ' OR 1=1 --. These payloads are submitted through each form, and the server responses are checked using regular expressions for typical database error messages. If patterns like "SQL syntax" or "mysql" are found in the response, the form is marked as potentially vulnerable to SQL Injection.
6. Implementing the CSRF Check: For CSRF detection, the program looks at each form and checks whether any input field name contains words such as "csrf" or "token". If a POST form does not contain such a token, it is reported as possibly missing CSRF protection. This is only a doubt; it highlights that the site or the URL needs further security checks. Finally, a simple text-based menu was added so that the user can choose between crawling, individual tests (XSS, SQLi, CSRF), or running a full scan. This makes the tool easier to demonstrate and use during practical sessions or project evaluation.

CONCLUSION: Thus, a python-based vulnerability scanner is designed that scans web applications for potential vulnerabilities. This scanner can be developed further to scan other vulnerabilities and external links using advanced and sophisticated programming for a real-time application.

