A Comparative Study of Automated and Manual VAPT Techniques for Web Application Security Assessment.

Target : BWAPP[Buggy Web Application]

1. Overview:

During the testing of BWAPP, we focused on identifyingvulnerabilities related to SQL injection, brute force, andLFI attacks. Our approach involved using both automatedtools and manual techniques to identify potentialvulnerabilities.

2. List of teammates participated in the HUFFPOST:

S.No	Name	Designation	Mobile No.
1.	Avula Bhumika	Team Leader	8688099414
2.	Guntuku Durga Surendra Kumar	Team Member	8179464989
3.	Avvaru Lokesh	Team Member	8074814537
4.	Jujjuvarapu Nagendra	Team Member	6300224187
5.	Chinta Sandeep	Team Member	9505543325

3.List of Vulnerable Parameter, Location discovered:

S.No	Vulnerability path	Name of the vulnerability	Reference CWE
1	http://10.0.2.5:8080/003/xss_post.php	XSS	CWE-79
2	http://10.0.2.4/001/vulnerabilities/sqli/	SQL INJECTION	CWE-89
3	http://10.0.2.5/003/html_post.php	HTML INJECTION	Web functionality(such as web widget)

4. Other Information

Tools Used :

1. Kali Linux
2. FireFox Browser

5 . Main vulnerability representation format :-

1— 1.1 . Vulnerability Name: Cross-Site Scripting (Reflected)

CWE : CWE-79

OWASP Category: A03:2021 – Injection

Description: Untrusted data enters a web application, typically from a web request

Business Impact: The application stores dangerous data in a database, message forum, visitor log, or other trusted data store. At a later time, the dangerous data is subsequently read back into the application and included in dynamic content. From an attacker's perspective, the optimal place to inject malicious content is in an area that is displayed to either many users or particularly interesting users. Interesting users typically have elevated privileges in the application or interact with sensitive data that is valuable to the attacker. If one of these users executes malicious content, the attacker may be able to perform privileged operations on behalf of the user or gain access to sensitive data belonging to the user. For example, the attacker might inject XSS into a log message, which might not be handled properly when an administrator views the logs.

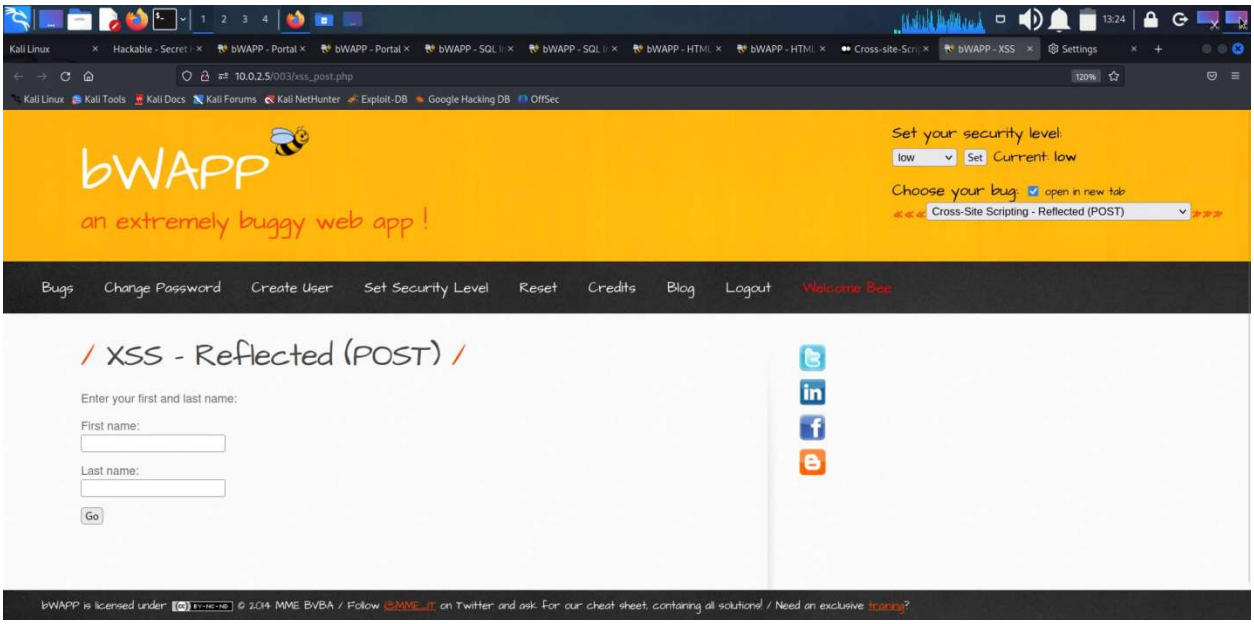
Vulnerability Path :http://10.0.2.5:8080/

Vulnerability Parameter:

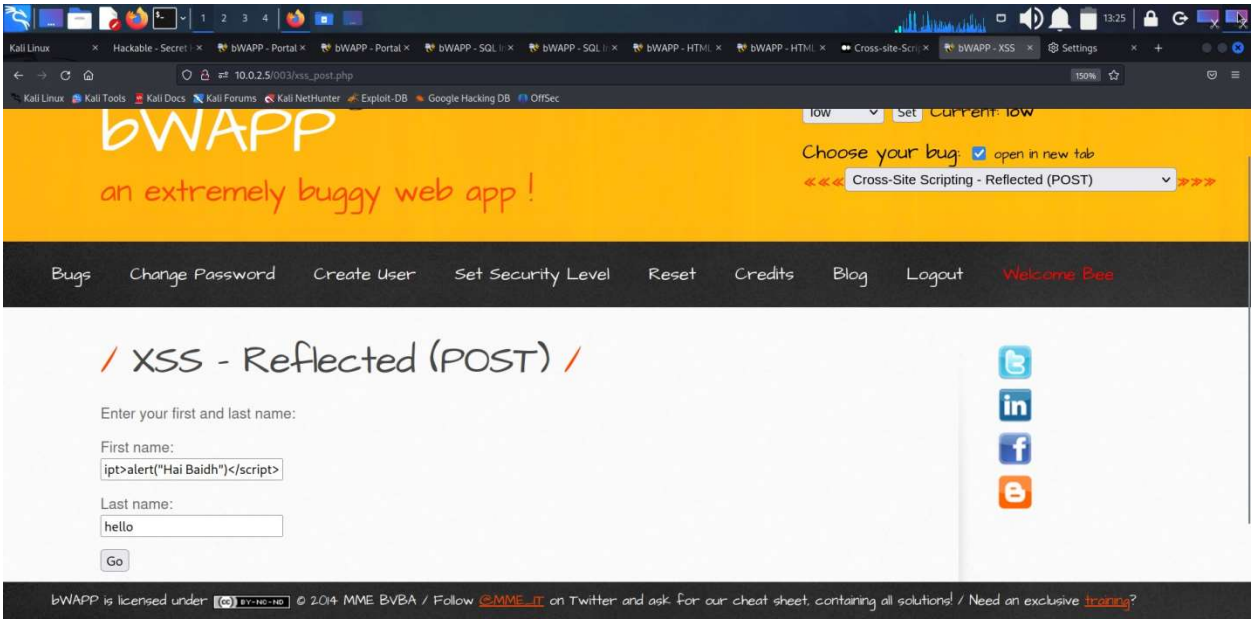
http://10.0.2.5:8080/003/xss_post.php

Steps to Reproduce :

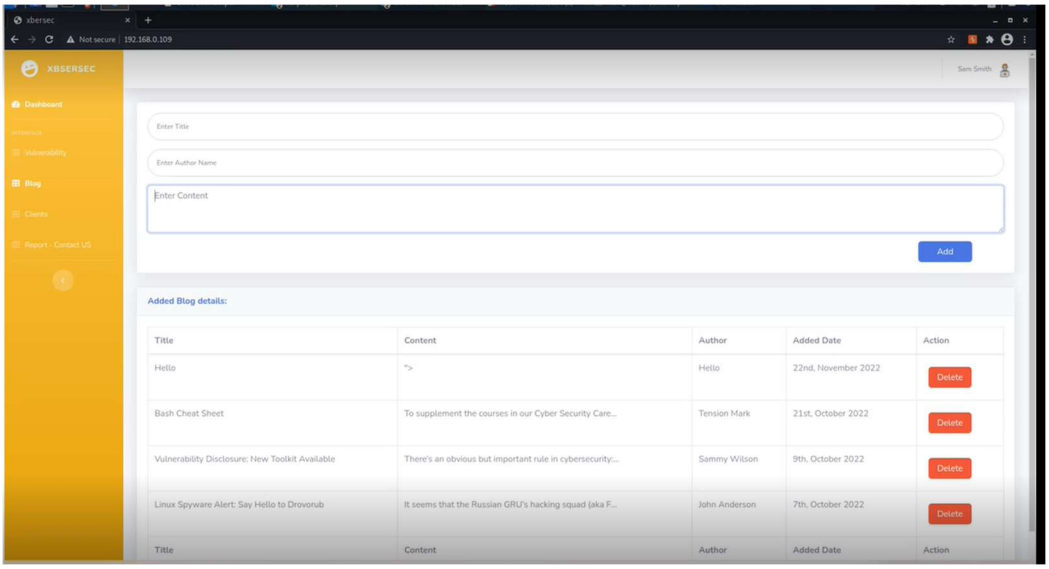
Step 1. Access the URL



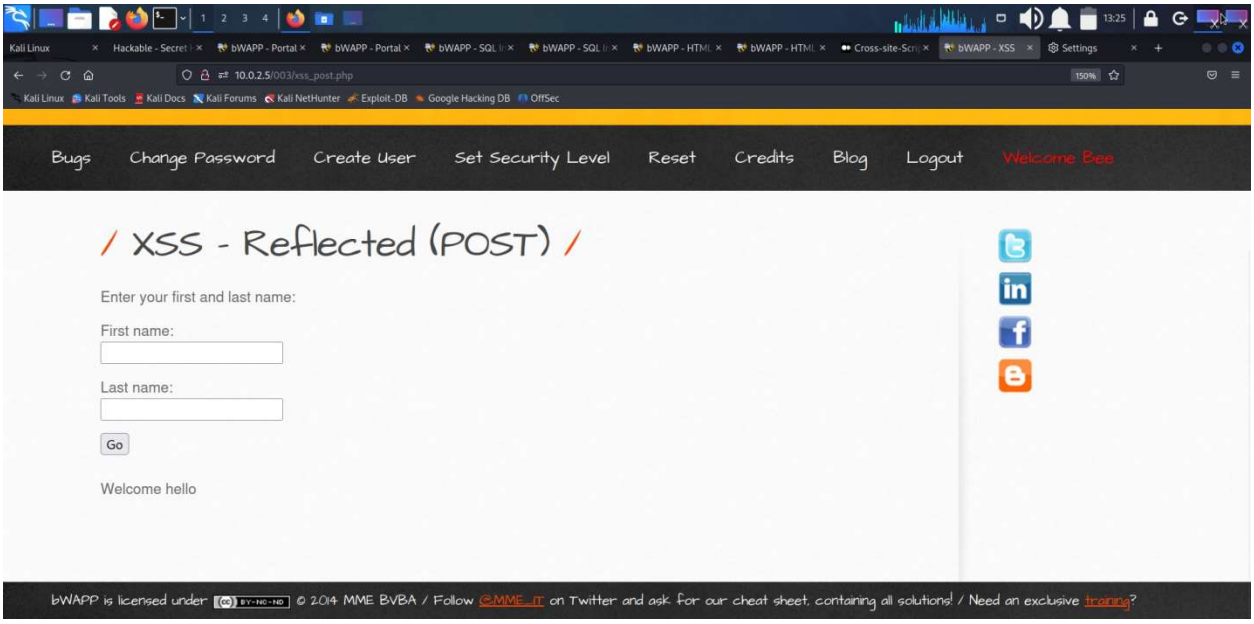
Step 2: Go to the login page and enter credentials along with a xsspayload



Step 3: Now there will be a pop up which is given in our xss payload.



Step 4:-we can also see the text below the inputs.



Recommendation:

- Note that proper output encoding, escaping, and quoting is the most effective solution for preventing XSS, although input validation may provide some defense-in-depth.

Vulnerability Name:-SQL Injection

CWE : -CWE-89

OWASP Category:-A03:2021 – Injection

Description:-SQL injection (SQLi) is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database

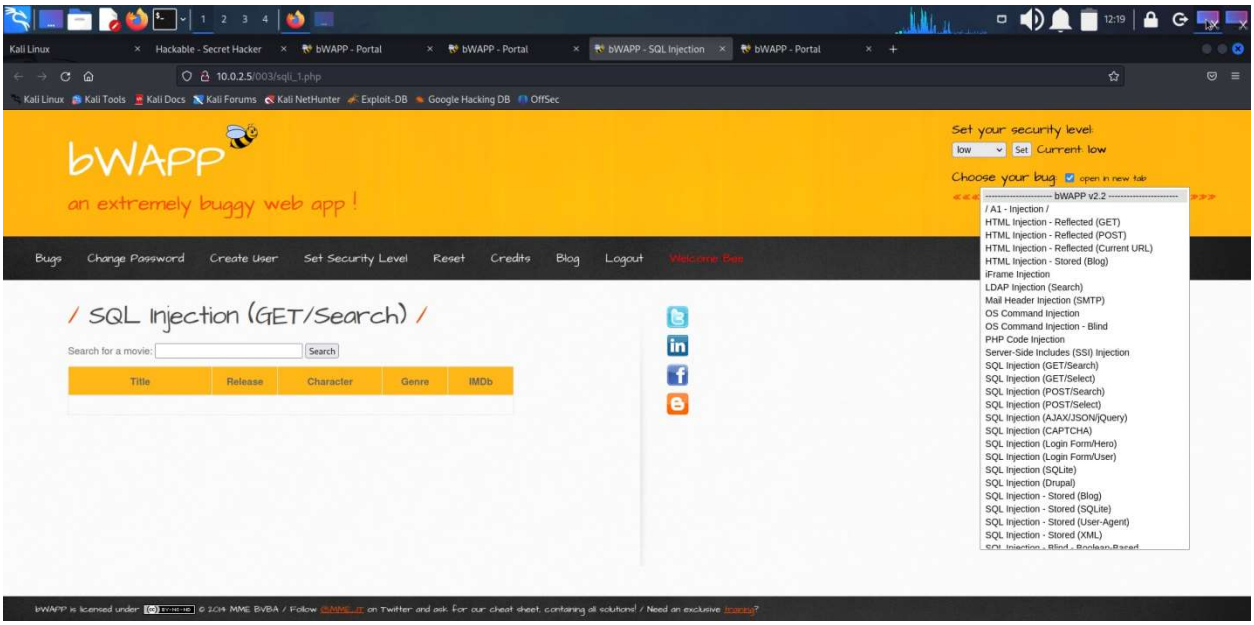
Business Impact::-A SQL injection attack consists of insertion or "injection" of a SQL query via the input data from the client to the application. A successful SQL injection exploit can read sensitive data from the database, modify database data (Insert/Update/Delete), execute administration operations on the database (such as shutdown the DBMS), recover the content of a given file present on the DBMS file system and in some cases issue commands to the operating system. SQL injection attacks are a type of injection attack, in which SQL commands are injected into dataplane input in order to affect the execution of predefined SQL commands. SQL injection attacks allow attackers to spoof identity, tamper with existing data, cause repudiation issues such as voiding transactions or changing balances, allow the complete disclosure of all data on the system, destroy the data or make it otherwise unavailable, and become administrators of the database server. SQL Injection is very common with PHP and ASP applications due to the prevalence of older functional interfaces.

Vulnerability Path :http://10.0.2.4/001/

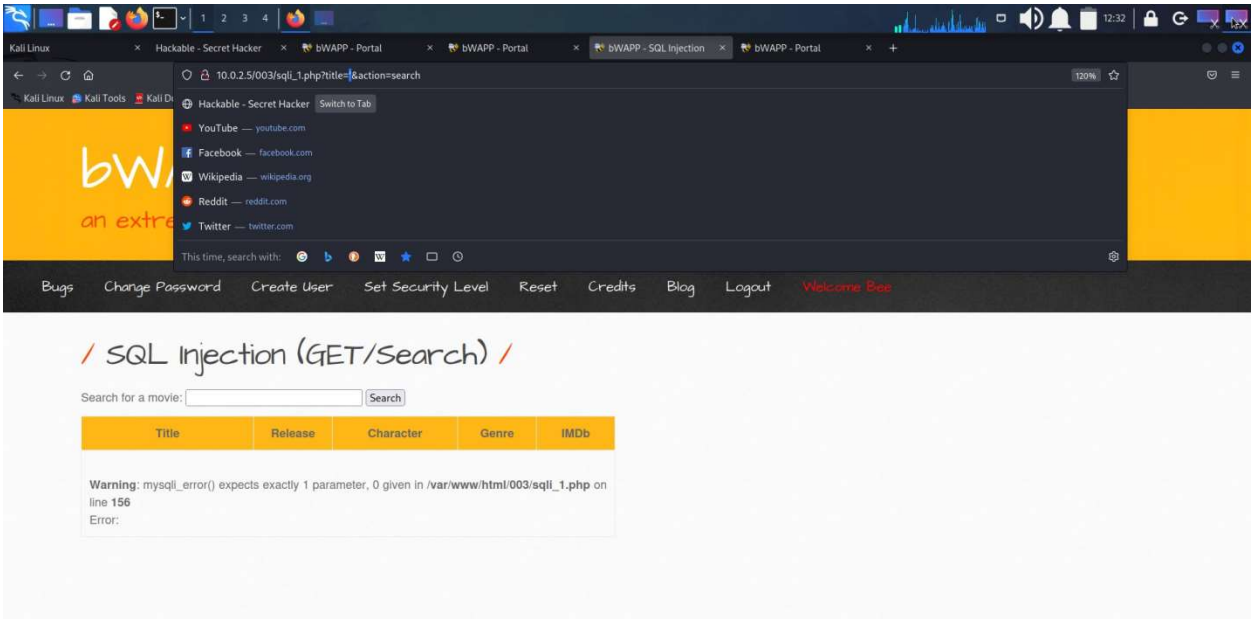
VulnerabilityParameter: http://10.0.2.4/001/vulnerabilities/sqli/

Steps to Reproduce :-

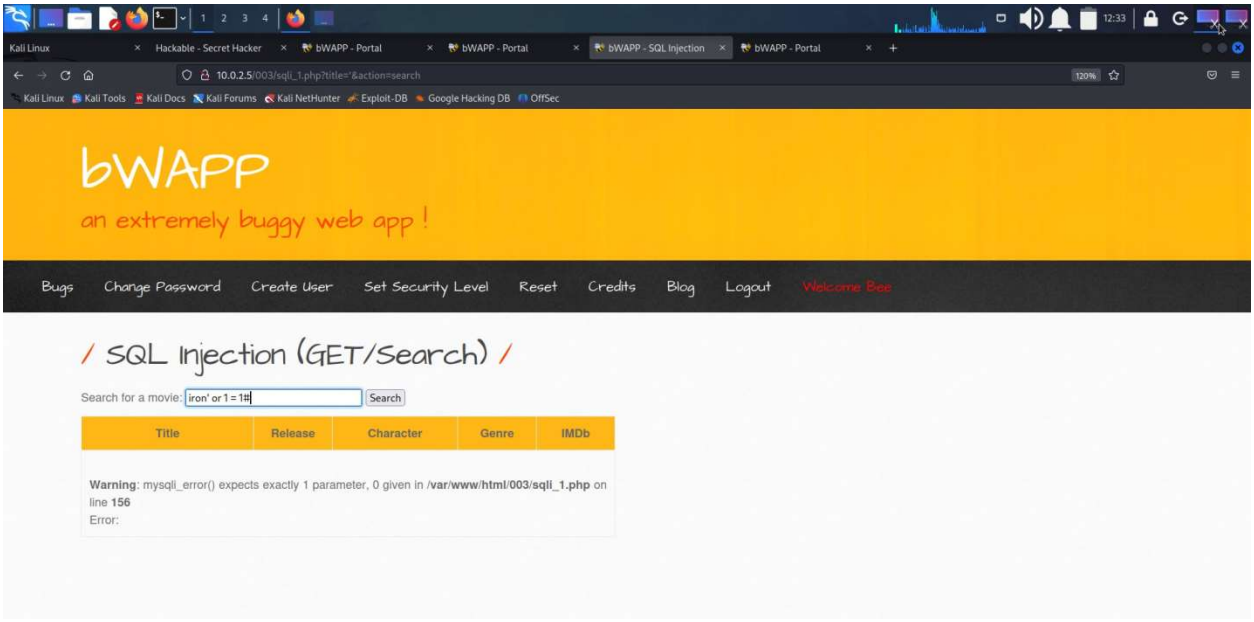
1.Access the Url



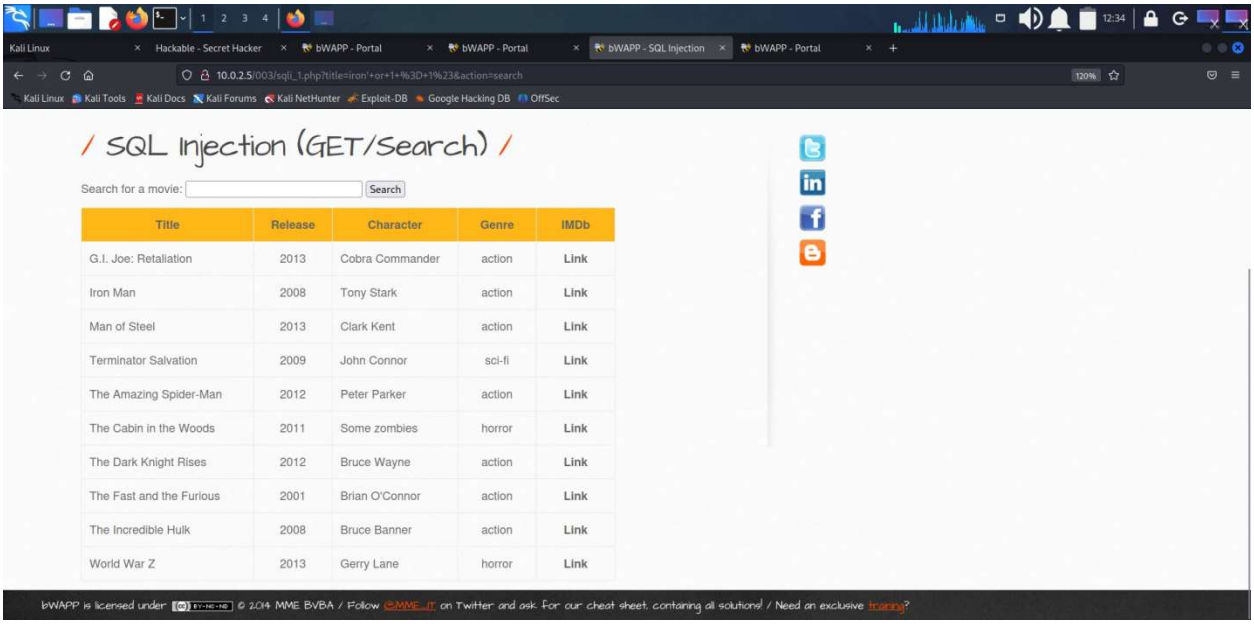
2.test for sqlerror using ‘ in title parameter



3.now give ansql payload as input



4.we received all the data



Recommendation:-

• Do not rely on client-side input validation

. ● Use a database user with restricted privileges.

● Use prepared statements and query parameterization.

● Scan your code for SQL injection vulnerabilities.

Vulnerability Name:-HTML Injection-Reflected(POST)

CWE : -CWE-79

OWASP Category:-A03:2021 – Injection

Description:-HTML Injection also known as Cross Site Scripting. It is a security vulnerability that allows an attacker to inject HTML code into web pages that are viewed by other users.

Business Impact:-

A phony form might be used by the attacker to steal password information saved in the browser or to fool a user into entering their login information. Malicious actors may be granted administrative access to the online application if the targeted user has those rights.

By carrying out an assault that is visible to the public, the attacker might seriously damage the image of the business, organization, or even nation. Users or clients may make poor judgments and lose faith in your cybersecurity procedures if a high-value page is vandalized or exploited to propagate misinformation.

HTML injection might be used as a technique by the attacker to progress to more severe assaults like CSRF.

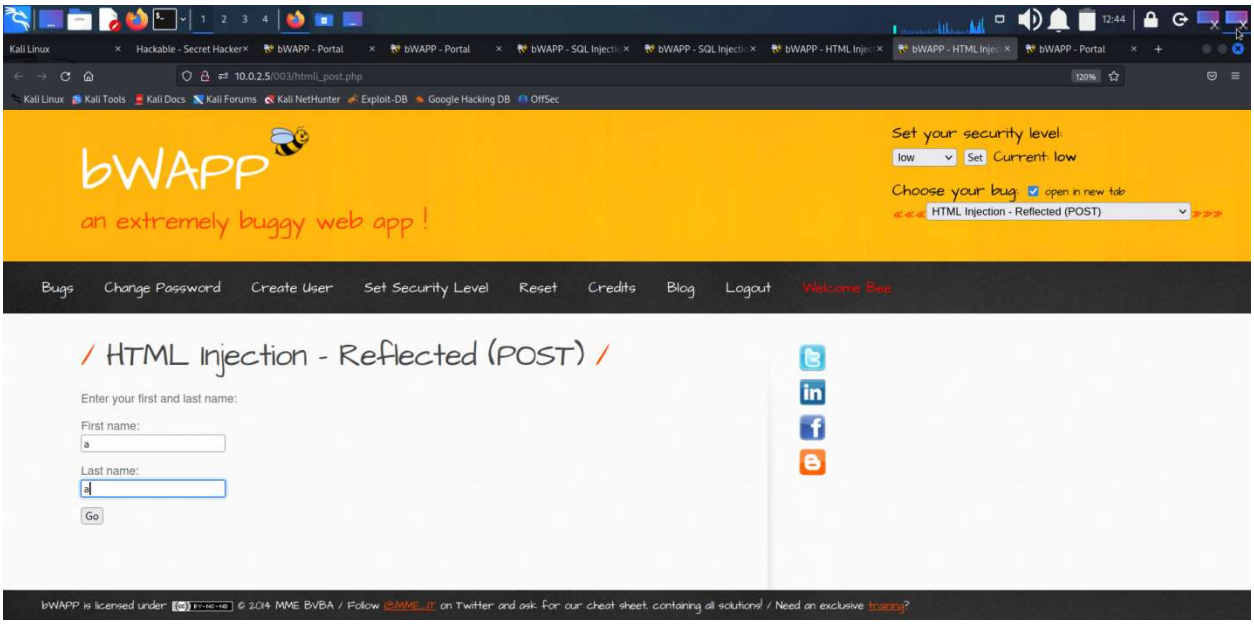
The attacker creates malicious links with his HTML content inserted into them, then emails the URLs to the victim. Because the page is hosted on a reputable domain, the user views it, which causes his identity to be stolen.

Vulnerability Path:http://10.0.2.5/003

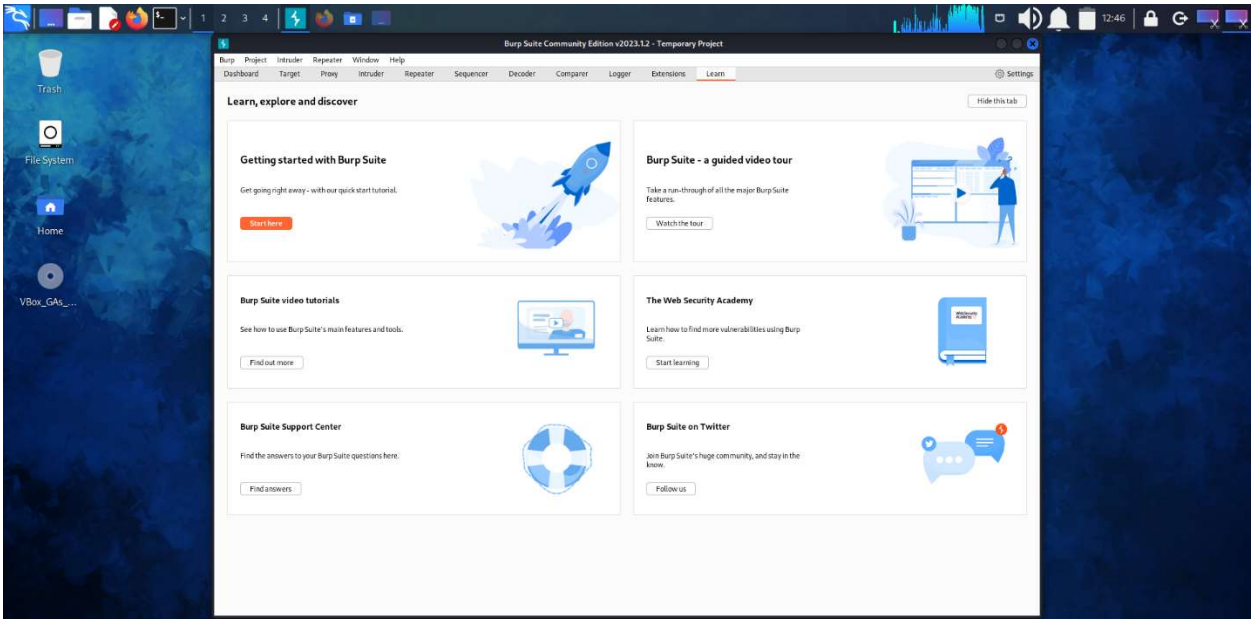
Vulnerability Parameter:-http://10.0.2.5/003/htmli_post.php

Steps to Reproduce :-

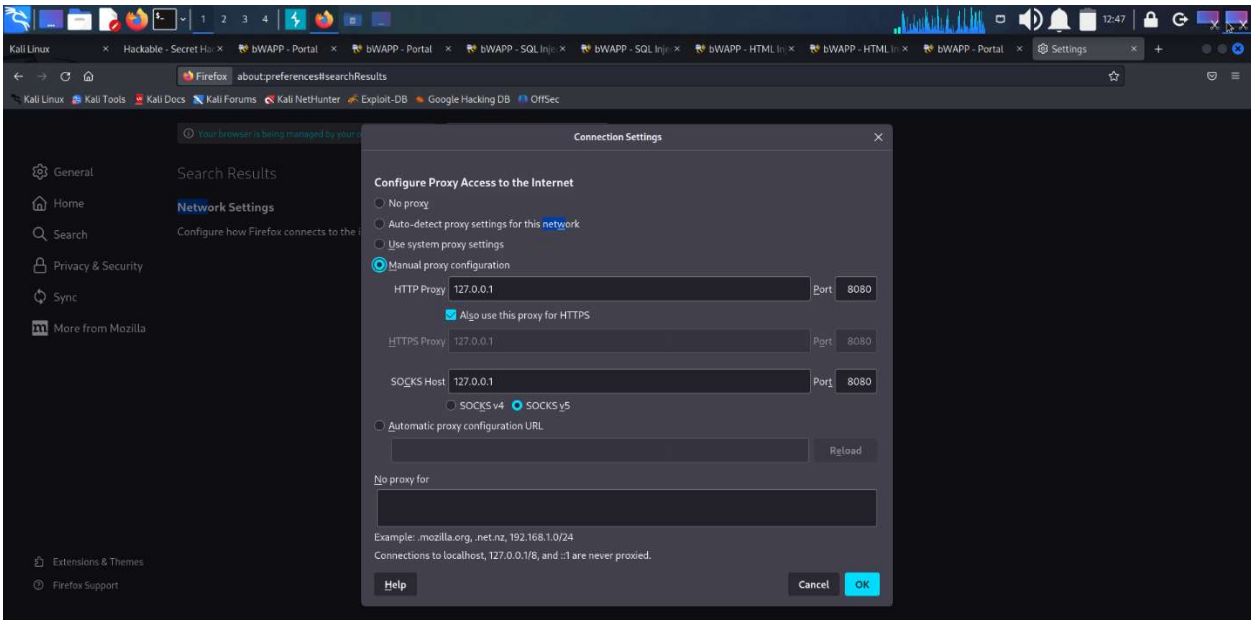
1.Access the url:



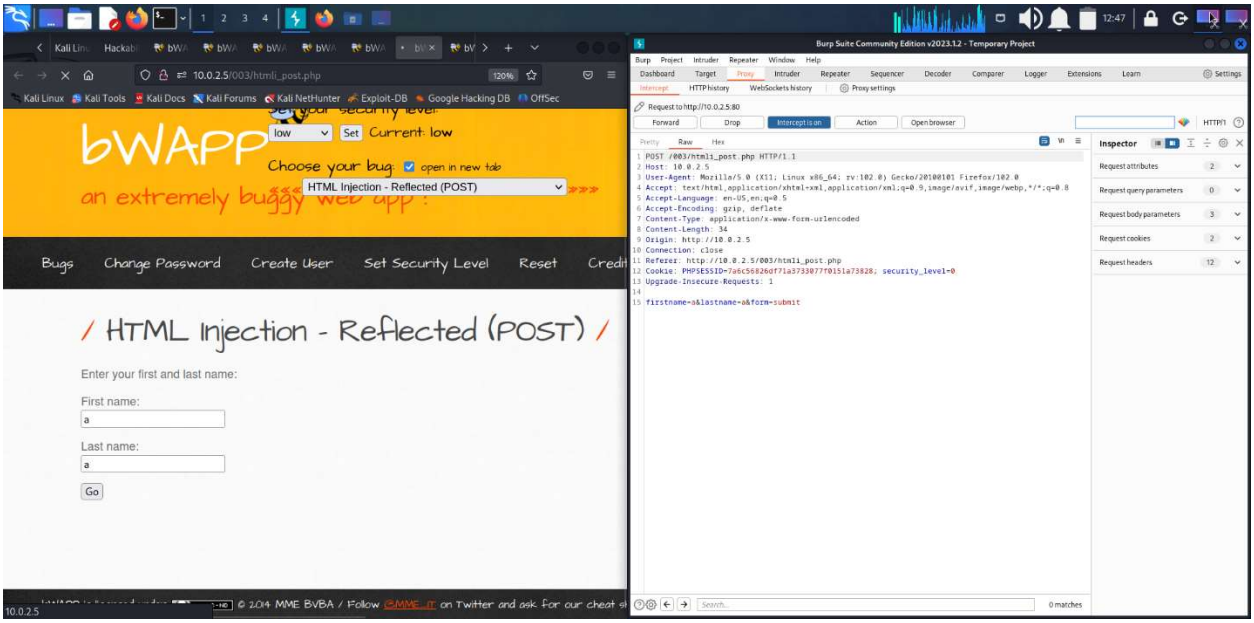
2.Now give some dummy data and open burp suite and wait for the request while making the proxy on.



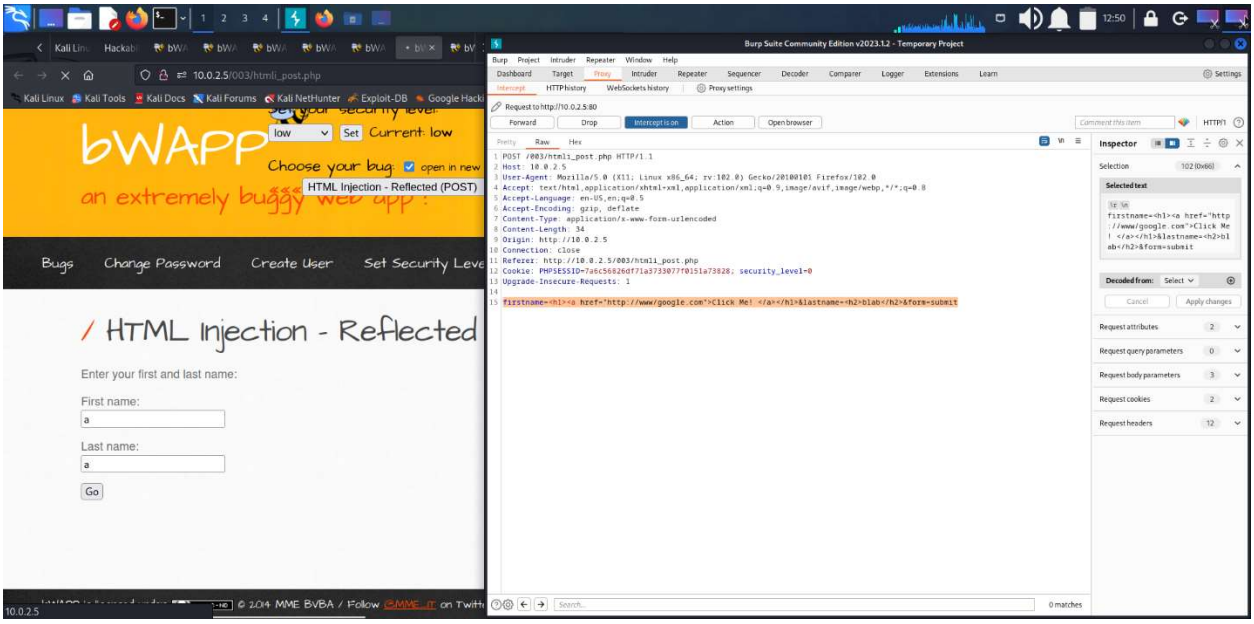
3.Also connect the browser to the local host proxy of the burpsuite.



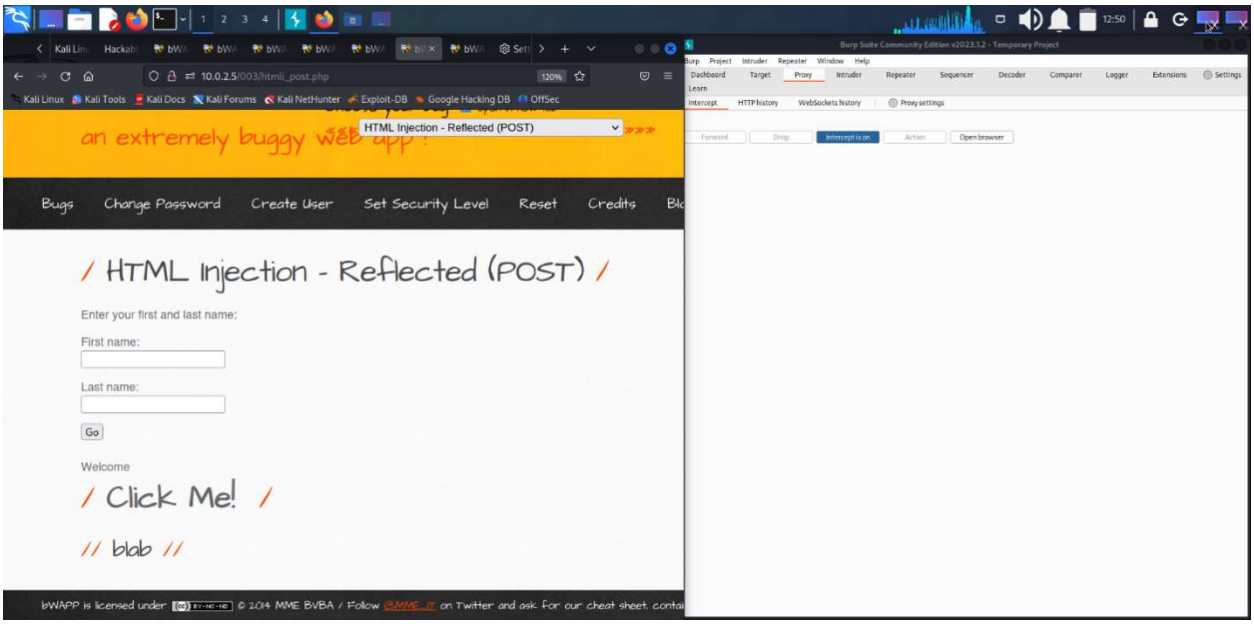
4.we successfully intercepted the request and can see the request data in raw form.



5.Now you can see the given first name and last name clearly and we inserted a anchor tag in between the first name value.



6.we successfully proved that there is no input validation ,as our input reflected on the site.



Recommendation:-

It is therefore essential to have appropriate data validation in place to prevent such attacks. Every input should be checked if it contains any script code or any HTML code. One should check, if the code contains any special script or HTML brackets – <script></script>, <html></html>.

The most common way of detecting HTML injection is by looking for HTML elements in the incoming HTTP stream that contains the user input. A naive validation of user input simply removes any HTML-syntax substrings (like tags and links) from any user-supplied text.